المناهج الدراسية السودانية

المرحلة الثانويــة

الصف الأول

علوم الحاسوب

الصف الثالث الثانوي

بسم الله الرحمن الرحيم

جمهورية السودان

وزارة التعليم العام

المركز القومى للمناهج والبحث التربوى

بخت الرضا

علوم الحاسوب

الصف الثالث الثانوي

الطبعة الثانية ٢٠٠٨م

إعداد : لجنة بتكليف من المركز القومي للمناهج والبحث التربوي مـــن الأساتذة :

أ . د : **عـــوض حـــاج علـــى** – مدير جامعة النيليـــن

دكتور : **السمانى عبد المطلب احمد** – جامعـــــة النيليـــن

الجمـــع بالحـــاسوب :

تهانى بابكر سليمان – المركز القومى للمناهج

نضال طلب احمد – جامعـــة النيليـــن

الأخراج الفنى والتصميم :

رحـــاب علـــى النقيب – جامعـــة النيليـــن

إبراهيم الفاضل الطاهر – المركز القومي للمناهج

الناشرون :

الطابعون :

رقم الإيداع: ٧٨٧/٢٠٠٨

المحتويات

مقدمة الكتاب

هذا هو المقرر الثالث في علوم الحاسوب بالمرحلة الثانوية، كان المقرر الأول بالسنة الأولى مقدمة عن الحاسوب ومكوناته واستخداماته، وكان المقرر الثاني بالسنة الثانية مقدمة في معالجة البيانات والبرمجة . أمّا المقرر الثالث الذي بين أيدينا يمثل المدخل إلى ما يعرف بعلوم الحاسوب البحتة وهي التصاميم والخوارزميات والبرمجيات التي تجعل من الحاسوب أداة فاعلة وميسرة لكل التطبيقات الهامة والتي تبدأ من تخزين البيانات واسترجاعها وتصنيفها وترتيبها وتأمينها ثم تيسير استخدام الحاسوب ونعني به نظم التشغيل و أخيراً التعرف علي الحاسوب كآله ذكية من خلال ما يعرف بالذكاء الاصطناعي .

لقد تم استخدام لغات "بسكال" و "برلوق" في التطبيقات المصاحبة للمقرر ولكن ليس المطلوب معرفة تفاصيل وقواعد هذه اللغات، و إنما المطلوب التعبير عن الخوارزمية بعبارات مشابهة لتلك اللغات ولقد أوضحنا ذلك من خلال شرحنا للبرامج باللغة العربية .

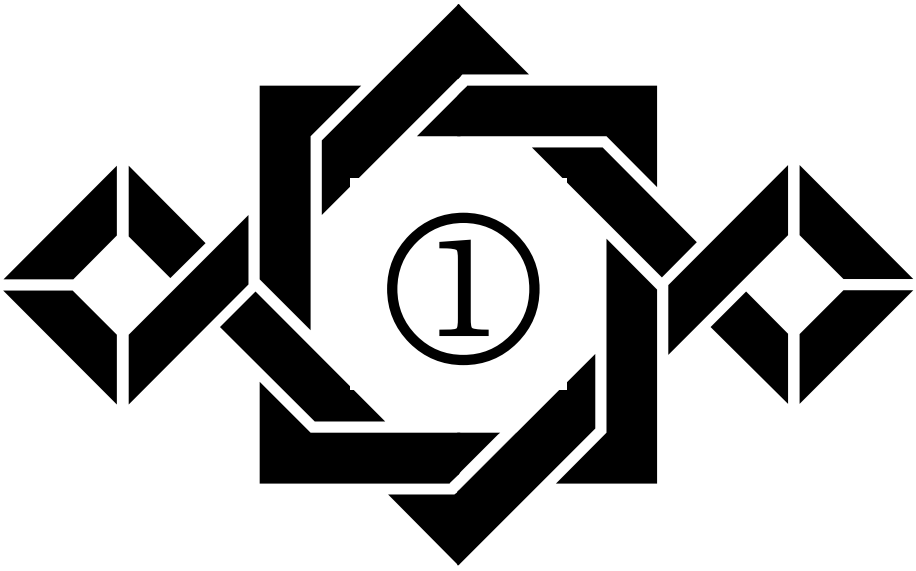
يهدف هذا المقرر إلى تعريف الطالب تعريفاً أكثر دقة في علوم الحاسوب وتقنية المعلومات حتى يستطيع التقديم لهذه التخصصات في المرحلة الجامعية وهو علي بينة من أمره . ولكن لابد من تنبيه الطالب أن هذه العلوم أي علوم الحاسوب هي علوم العصر التي أصبحت تشكل عنصراً أساسياً لكل

العلوم التطبيقية والتقنية الأخرى مثل الهندسة والعلوم والطب والزارعة والعلوم البيطرية والعلوم الإدارية والاقتصادية . . . . . الخ .

لقد وضعنا تمريناً في نهاية كل باب يمثل ما نريد من الطالب فهمه واستيعابه وسيكون الامتحان مشابهاً لهذه التمارين بإذن الله تعالى .

نرجو من أبنائنا الطلاب الولوج إلى هذه المعرفة الحية واختيار هذه المادة التي تعتبر من المواد المؤهلة للقبول في كل الكليات الجامعية تقريباً .

المؤلفان

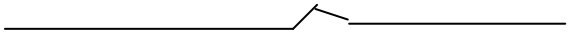


الدّوائر المنطقيّة والعدّ الثنائيّ

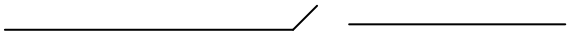
الدوائر المنطقية والعد الثنائي

١. مدخل :

عند تعريفنا للحاسوب قلنا إنه جهاز الكتروني يعامل المعلومات وهي في صورة رقمية ثنائية تمثل منطقياً بصفر وواحد ، وفيزائياً بوجود تيار كهربي وعدم وجود تيار كهربائي ، أو بلغة أخرى أن يكون المفتاح الكهربائي في وضع موصل للتيار الكهربائي وفي وضع فاصل للتيار الكهربائي كما في الرسم :



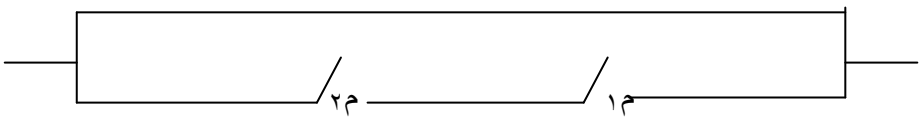
المفتاح في حالة موصل للتيار الكهربائي



المفتاح في حالة فاصل للتيار الكهربائي

بالطبع لا يمكن أن يكون المفتاح في الحالتين معاً في وقت واحد لذا سنرمز للمفتاح إذا كان في حالة موصل للتيار بالرمز م وفي حالة فاصل للتيار بالرمز $\overline{م}$ ويقابل هاتين الحالتين الرقم واحد في حالة م أي موصل التيار ، والرقم صفر في حالة $\overline{م}$ أي فاصل التيار .

إذا كان لدينا مفتاحان م١ و م٢ موصلان على التوالي كما في الرسم :

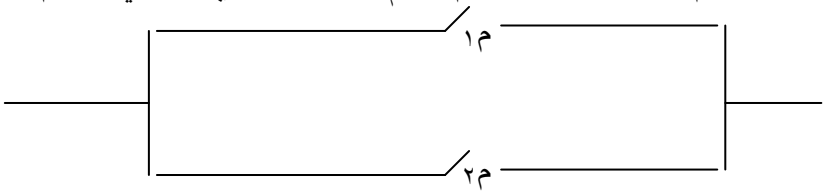


فإن التيار كما نعلم في دروس الفيزياء لا يمر إلا إذا كان المفتاحان موصّلين أما إذا كان أحد المفتاحين في حالة فاصل التيار، فإن ناتج هذه الدائرة الكهربية سيكون انقطاع التيار . يمكن تمثيل هذه الدائرة منطقياً بالجدول التالي :

م١	م٢	الدائرة الكهربية
١	١	١
١	٠	٠
٠	١	٠
٠	٠	٠

هذه الدائرة تطابق العلاقة م١ ∧ م٢ في المنطق الجبري وتعني ان هذه العلاقة م١ ∧ م٢ لا يمكن أن تكون حقيقية إلا إذا كانت م١ حقيقية وم٢ حقيقية في آن واحد . (هذه العلاقة تعرف بعلاقة "و" في المنطق الرياضي وعلاقة "×" في الجبر).

أمّا إذا تم توصيل المفتاحين م١ و م٢ على التوازي كما في الرسم :



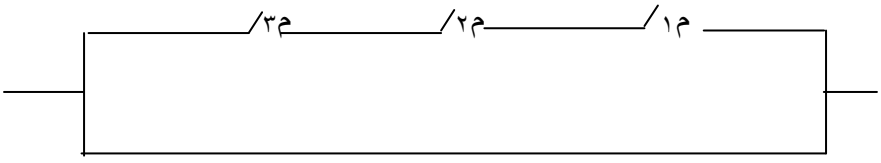
فإنه إذا مر التيار بأحد المفتاحين فإن ناتج الدائرة الكهربية سيكون اتصال التيار ويمكن تمثيل هذه الدائرة منطقياً بالجدول التالي :

م١	م٢	الدائرة الكهربية
١	١	١
١	٠	١
٠	١	١
٠	٠	٠

هذه الدائرة تطابق العلاقة م١∨م٢ في المنطق الرياضي أي أن ناتج م١ ∨م٢ يكون حقيقة إذا كان م١ أو م٢ حقيقة (هذه العلاقة تعرف بعلاقة "أو" في المنطق الرياضي وعلاقة "+" في الجبر) مع ملاحظة أن ١ + ١ = ١ اي وجود التيار من جهتين لا يؤثر في حقيقة وجود التيار) .

مثال (١) :

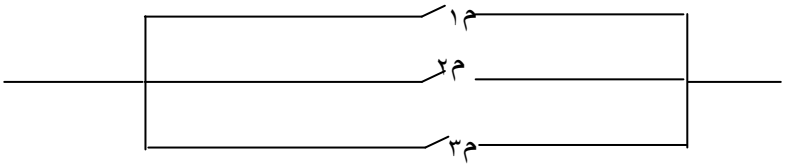
إذا كان لدينا ثلاث مفاتيح م١ و م٢ وم٣ ، في دائرة موصلة على التوالي كما في الرسم فما ناتج هذه الدائرة :



ناتج هذه الدائرة هو (م١ ∧ م٢) ∧ م٣ = م١ ∧ م٢ ∧ م٣

مثال (٢) :

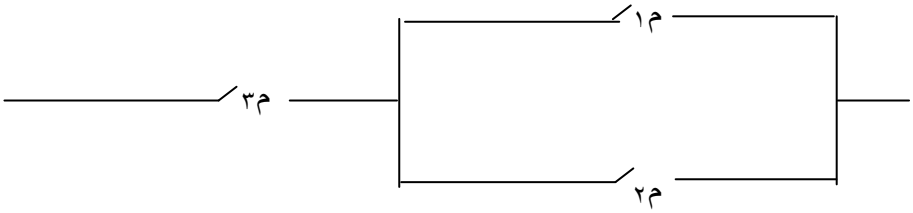
أما إذا كانت مفاتيح م١ و م٢ وم٣ موصلة على التوازي كما في الرسم :



فإن ناتج الدائرة يطابق (م١ ∨ م٢) ∨ م٣ = م١ ∨ م٢ ∨ م٣ .

مثال (٣) :

كذلك يمكن أن تكون م١ و م٢ وم٣ موصلة على النحو التالي :



وهذه تطابق المكافئ المنطقي (م١ ∨ م٢) ∧ م٣ وبالجبر (م١ + م٢) × م٣

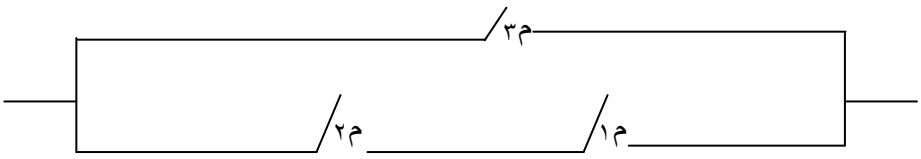
قاعدة ١

(م١ ∨ م٢) ∧ م٣ = (م١∧م٣) ∨ (م٢∧م٣) أو

(م١ + م٢) × م٣ =(م١ × م٣) + (م٢ × م٣)

مثال (٤) :

كذلك يمكن أن تكون م١ و م٢ وم٣ موصلة على النحو التالي :



وهذه تطابق المكافئ المنطقي (م١∧م٢) ∨م٣) وبالجبر (م١ × م٢) + م٣

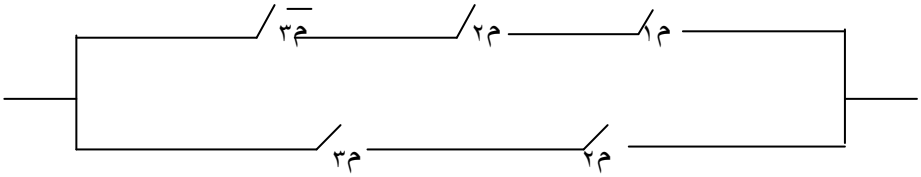
قاعدة ٢

$(م_1 \wedge م_2) \vee م_3 = (م_1 \vee م_3) \wedge (م_2 \vee م_3)$ أو

$(م_1 \times م_2) + م_3 = (م_1 + م_3) \times (م_2 + م_3)$

مثال (٥) :

عبر بالجبر المنطقي عن الدائرة :



هذه تطابق المكافئ المنطقي $(م_1 \wedge م_2 \wedge \overline{م_3}) \vee (م_2 \wedge م_3)$ [قاعده ١]

$= \{(م_1 \wedge م_2 \wedge \overline{م_3}) \vee م_2\} \wedge \{(م_1 \wedge \overline{م_2} \wedge م_3) \vee م_3\}$

وبالجبر $= م_1 \times م_2 \times \overline{م_3} + م_2 \times م_3$

$= (م_1 \times م_2 \times \overline{م_3} + م_2) \times (م_1 \times م_2 \times \overline{م_3} + م_3)$.

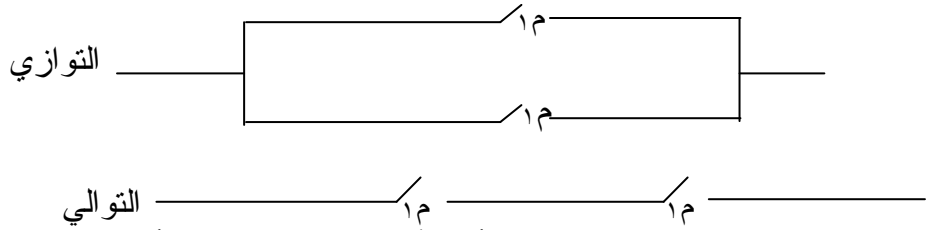
يمكن تمثيل الدائرة الكهربية لهذا المثال منطقياً بجدول الصواب التالي :

م١	م٢	م٣	$\overline{م_3}$	$م_1 \wedge م_2 \wedge \overline{م_3}$	$م_2 \wedge م_3$	الدائرة الكهربية
١	١	١	٠	٠	١	١
١	١	٠	١	١	٠	١
١	٠	١	٠	٠	٠	٠
١	٠	٠	١	٠	٠	٠
٠	١	١	٠	٠	١	١
٠	١	٠	١	٠	٠	٠
٠	٠	١	٠	٠	٠	٠
٠	٠	٠	١	٠	٠	٠

لرسم جدول هذه الدائرة بدأنا بتحديد الاحتمالات الثمانية لظروف فتح المفاتيح أو قفل المفاتيح الثلاث ، ثم عملنا عموداً لعكس حالة المفتاح م٣ لوجوده معكوساً في موقع آخر في الدائرة الكهربية ، بعد ذلك ربطنا المفاتيح التي على التوالي مع بعضها واستخدمنا الجبر لعلاقة الضرب ، ثم ربطنا المجموعتين بعلامة الجمع لأنهما مربوطتان على التوازي .

٣. قوانين هامة في الدوائر المنطقية :

قانون (١) :



تكرار نفس المفتاح لا يعني شيئاً أي كأنه مفتاح واحد سواء أكان ذلك على التوالي أم على التوازي بهذا يكون :

أ . على التوالي :

م١ ∧ م١ = م١	أو	م١ × م١ = م١

إذا كان م = ١ فإن م١ × م١ = ١ وإذا كان م١ = ٠ فإن م١ × م١ = ٠

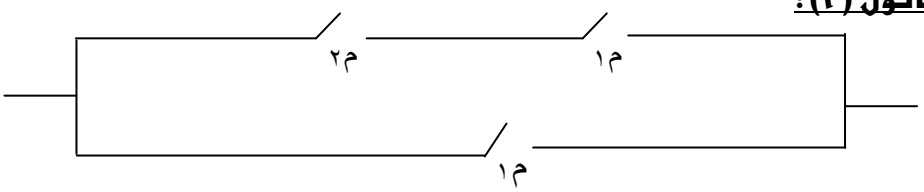
أي في كلتا الحالتين م١ × م١ = م١

ب. على التوازي :

م١ ∨ م١ = م١	أو	م١ + م١ = م١

فإذا كان م١ + م١ = ١ + ١ = ١ (فيزيائياً) أما إذا كان م١ = ٠ فإن م١ + م١= ٠

قانون (٢) :

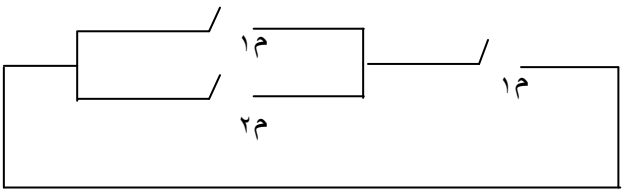


وجود نفس المفتاح على التوالي مع مفتاح آخر على التوازي يكون هو الحاكم أي

م١ ∨ (م١ ∧ م٢) = م١	أو	م١ + (م١ × م٢) = م١

وهذه واضحة إذ أنه إذا كان م١ = صفراً فإن القيمة تساوي صفراً حتى إذا كان م٢ واحداً وكذلك إذا كان م١ واحداً فإن القيمة واحد حتى إذا كان م٢ = صفراً .

قانون (٣):

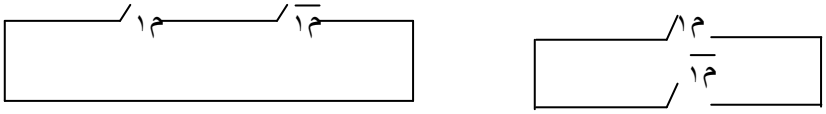


إذا كان م١ وم٢ على التوازي ثم م١ يتصل بهما مرة أخرى على التوالي فإن الحاكم هو م١ أي

م١ ∧ (م١ ∨ م٢) = م١ أو م١ × (م١ + م٢) = م١

لأنه إذا كان م١= ١ فإن النتيجة واحد حتى ولو كان م٢ = صفر وكذلك إذا كان م١ = صفرفإن النتيجة صفر حتى ولو كان م٢ = واحداً .

قانون (٤):



إذا كان م١ و$\overline{م_١}$ موصلة على التوالي فإن النتيجة في كل الأحوال فاصل التيار أي القيمة المنطقية . م١ ∧ $\overline{م_١}$ = ٠ وأما إذا كانت موصولة على التوازي فإن القيمة في كل الأحوال موصل التيار أي م١ ∨ $\overline{م_١}$= ١.

م١ ∧ $\overline{م_١}$ = ٠ ، م١ ∨ $\overline{م_١}$= ١

٣. اختصار الدوائر المنطقية :

يمكن باستخدام القوانين السابقة اختصار الدائرة الكهربية. ففي المثال السابق (مثال (٥)) نجد أن :

$$(م_1 \wedge م_2 \wedge \overline{م_3}) \vee (م_2 \wedge م_3)$$

$$= (م_1 \vee (م_2 \wedge م_3)) \wedge (م_2 \vee (م_2 \wedge م_3)) \wedge (\overline{م_3} \vee (م_2 \wedge م_3)) \quad \text{[قاعدة ٢]}$$

$$= (م_1 \vee (م_2 \wedge م_3)) \wedge (م_2) \wedge (\overline{م_3} \vee (م_2 \wedge م_3)) \quad \text{[قانون ٣]}$$

$$= (م_1 \vee (م_2 \wedge م_3)) \wedge (م_2) \wedge ((\overline{م_3} \vee م_2)) \wedge (\overline{م_3} \vee م_3)) \quad \text{[قاعدة ٢]}$$

$$= (م_1 \vee (م_2 \wedge م_3)) \wedge (م_2) \wedge ((\overline{م_3} \vee م_2)) \wedge (1)) \quad \text{[قاعدة ٢]}$$

$$= (م_1 \vee (م_2 \wedge م_3)) \wedge (م_2 \wedge (\overline{م_3} \vee م_2)$$

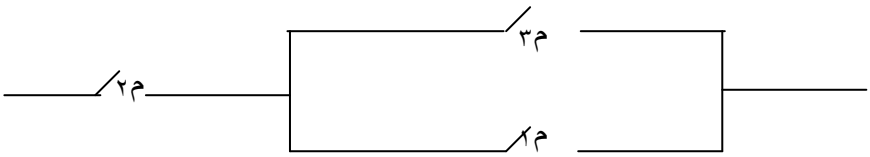
$$= (م_1 \vee (م_2 \wedge م_3)) \wedge (م_2) \quad \text{[قانون ٣]}$$

$$= ((م_1 \vee م_2) \wedge م_2) \wedge ((م_1 \vee م_3) \wedge م_2) \quad \text{[قاعدة ٢]}$$

$$= م_2 \wedge (م_1 \vee م_3) \wedge م_2 = (م_2 \wedge م_2) \wedge (م_1 \vee م_3)$$

$$= م_2 \wedge (م_1 \vee م_3) \quad \text{[قانون ١]}$$

وعليه تكون الدائرة :

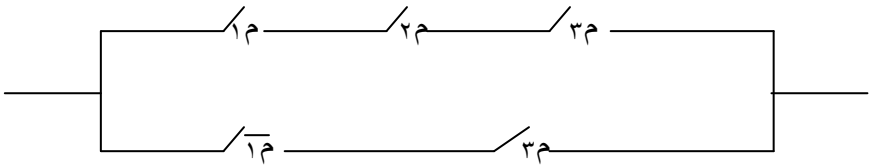


يمكن تمثيل الدائرة الكهربية لهذا المثال منطقياً بجدول الصواب التالي :

م١	م٢	م٣	$م_1 \vee م_3$	**الدائرة الكهربية** $= م_2 \wedge (م_1 \vee م_3)$
١	١	١	١	١
١	١	٠	١	١
١	٠	١	١	٠
١	٠	٠	١	٠
٠	١	١	١	١
٠	١	٠	٠	٠
٠	٠	١	١	٠
٠	٠	٠	٠	٠

مثال (٦) :

عبر عن الدائرة في الشكل التالي بالمنطق الجبري ثم اختصرها .



يمكن التعبير عن هذه الدائرة منطقياً بالمكافئ المنطقي :

$(م_1 \wedge م_2 \wedge م_3) \vee (\overline{م_1} \wedge م_3)$

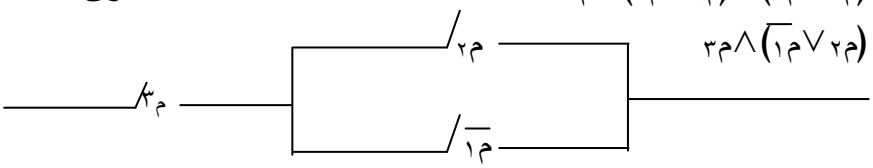
= $((م_1 \wedge م_2 \wedge م_3) \vee \overline{م_1}) \wedge (م_3 \vee (م_1 \wedge م_2 \wedge م_3))$ [قاعدة ٢]

= $((م_1 \wedge م_2 \wedge م_3) \vee \overline{م_1}) \wedge (م_3)$ [قانون ٣]

= $(م_1 \vee \overline{م_1}) \wedge (م_2 \vee \overline{م_1}) \wedge (م_3 \vee \overline{م_1}) \wedge م_3$ [قاعدة ٢]

= $(م_2 \vee \overline{م_1}) \wedge (م_3 \vee \overline{م_1}) \wedge م_3$ [قانون ٣]

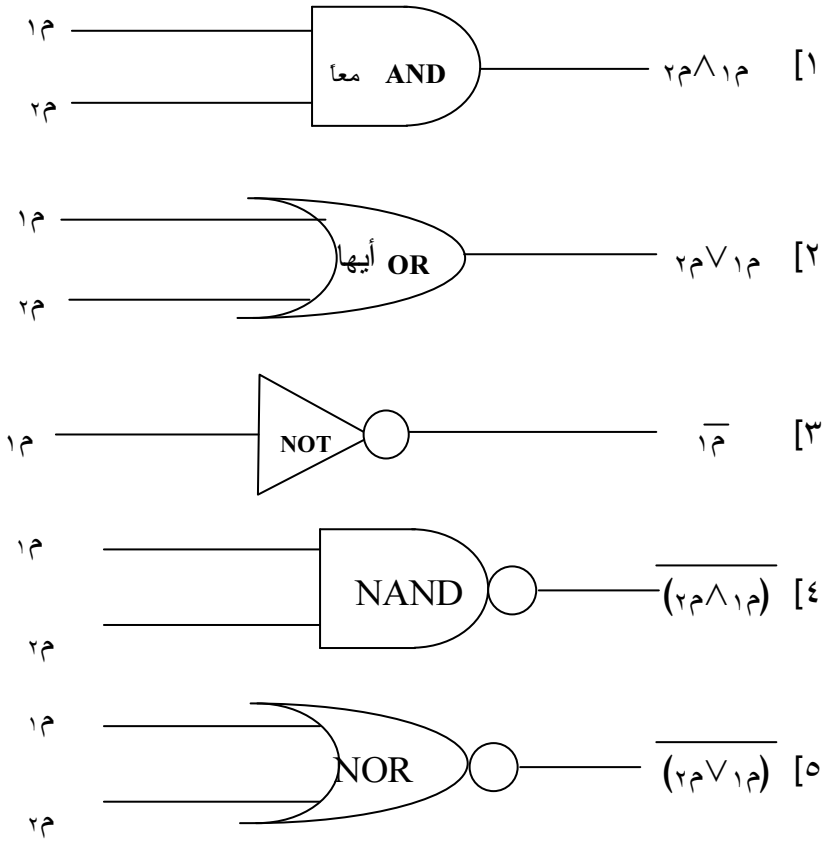
= $(م_2 \vee \overline{م_1}) \wedge م_3$



يمكن تمثيلها بجدول الصواب التالي :

م١	م٢	م٣	$\overline{م_1}$	$\overline{م_1} \vee م_2$	**الدائرة الكهربية** = $م_3 \wedge (\overline{م_1} \vee م_2)$
١	١	١	٠	١	١
١	١	٠	٠	١	٠
١	٠	١	٠	٠	٠
١	٠	٠	٠	٠	٠
٠	١	١	١	١	١
٠	١	٠	١	١	٠
٠	٠	١	١	١	١
٠	٠	٠	١	١	٠

٤. مصطلح الأشكال الهندسية في الدوائر المنطقية :

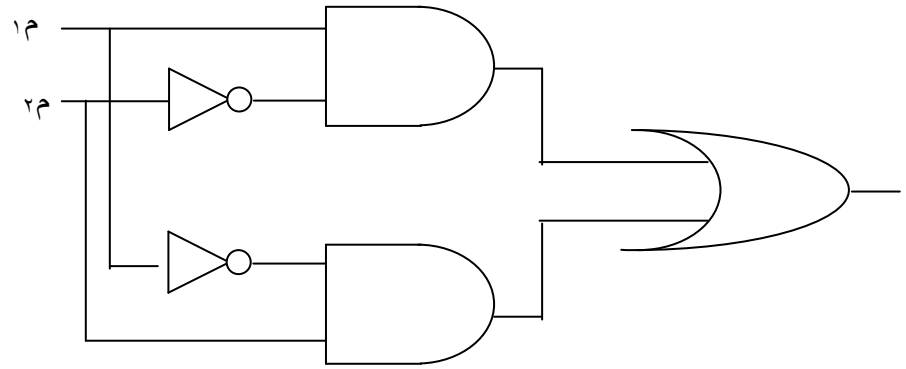


صمام NAND هو مكافي لصمام AND يتبعه صمام NOT أما صمام NOR هو يكافئ لصمام يتبعة صمام NOT وبالتالي يكون جدول الصواب لهما كالاتي :

م١	م٢	م١ ∧ م٢	م١ ∨ م٢	NAND	NOR
١	١	١	٠	٠	٠
١	٠	٠	١	١	٠
٠	١	٠	١	١	٠
٠	٠	٠	٠	١	١

٦] م١ ⊻ م٢ = (م١ ∧ $\overline{م٢}$) ∨ ($\overline{م١}$ ∧ م٢)

الأول مع معكوس الثاني أو معكوس الأول مع الثاني ويمكن رسمها بالشكل التالي :

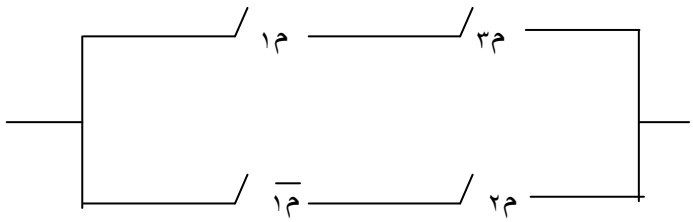


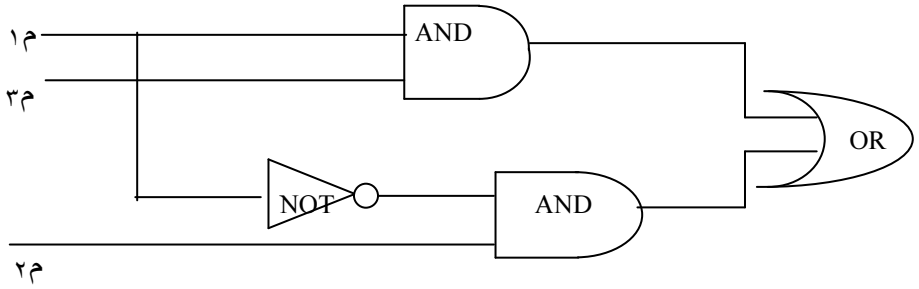
ويمكن تصميم جدول الصواب علي النحو التالي :

م١	م٢	$\overline{م١}$	$\overline{م٢}$	$\overline{م١}$ ∧ م٢	م١ ∧ $\overline{م٢}$	($\overline{م١}$ ∧ م٢) ∨ (م١ ∧ $\overline{م٢}$)
١	١	٠	٠	٠	٠	١
١	٠	٠	١	٠	١	١
٠	١	١	٠	١	٠	٠
٠	٠	١	١	٠	٠	٠

مثال (٧) :

عبر عن الدائرة المنطقية أدناه بأشكال المصطلح الهندسي. ثم جد جدول الصواب.

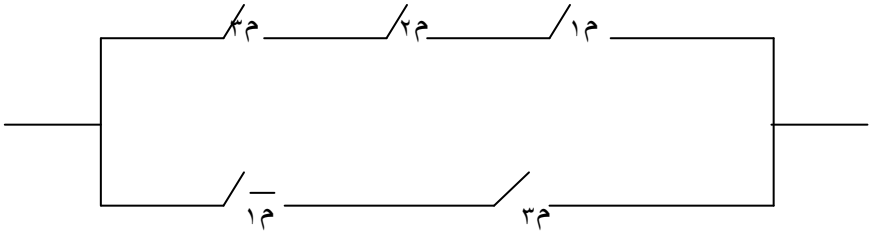


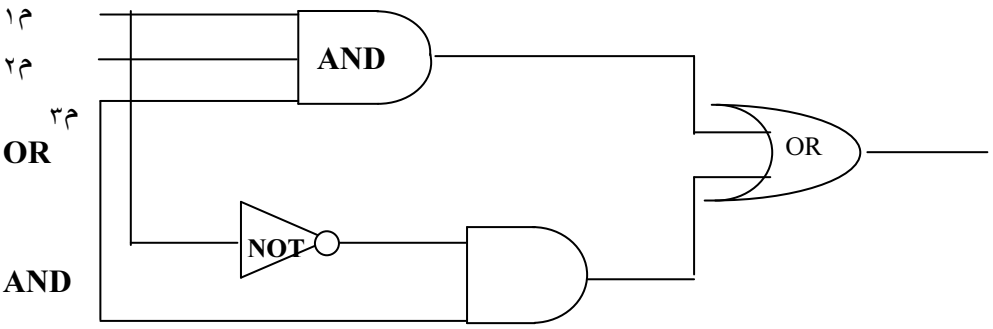


م١	م٢	م٣	$\overline{م١}$	م١ ∧ م٣	$\overline{م١}$ ∧ م٢	(م١ ∧ م3) ∨ ($\overline{م١}$ ∧ م٢)
١	١	١	٠	١	٠	١
١	١	٠	٠	٠	٠	٠
١	٠	١	٠	١	٠	١
١	٠	٠	٠	٠	٠	٠
٠	١	١	١	٠	١	١
٠	١	٠	١	٠	١	١
٠	٠	١	١	٠	٠	٠
٠	٠	٠	١	٠	٠	٠

مثال (٨) :

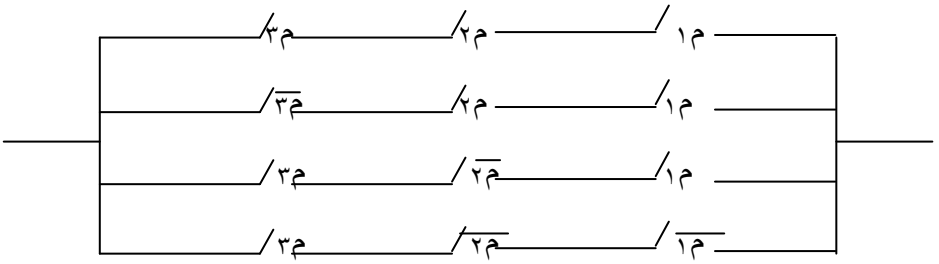
عبر عن الدائرة المنطقية الآتية بأشكال المصطلح الهندسي.





مثال (٩)

عبر عن الدائرة الآتية بأشكال المصطلح الهندسي .

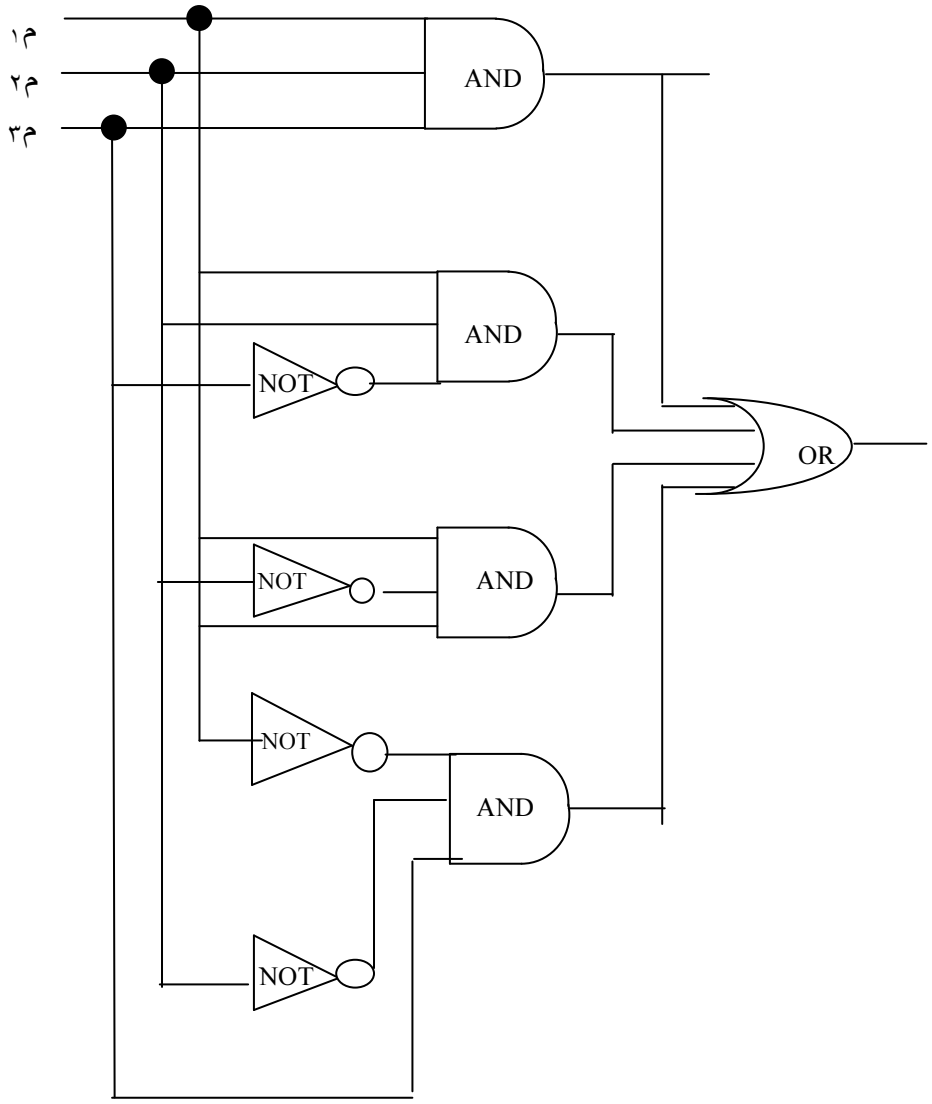


المكافئ المنطقي لهذه الدائرة :

(م١∧م٢∧م٣)∨(م١∧م٢∧م٣)∨(م١∧م٢∧م٣)∨(م١∧م٢∧م٣)

(الاول)∨ (الثاني)∨ (الثالث)∨ (الرابع)

م١	م٢	م٣	م١	م٢	م٣	الأول	الثاني	الثالث	الرابع	الدائرة
١	١	١	٠	٠	٠	١	٠	٠	٠	١
١	١	٠	٠	٠	١	٠	١	٠	٠	١
١	٠	١	٠	١	٠	٠	٠	١	٠	١
١	٠	٠	٠	١	١	٠	٠	٠	٠	٠
٠	١	١	١	٠	٠	٠	٠	٠	٠	٠
٠	١	٠	١	٠	١	٠	٠	٠	٠	٠
٠	٠	١	١	١	٠	٠	٠	٠	١	١
٠	٠	٠	١	١	١	٠	٠	٠	٠	٠


م١
م٢
م٣
AND
AND
NOT
OR
NOT
AND
NOT
AND
NOT

٥. النظام العددي الثنائي :

يسمى النظام العددي الذي يتعامل به في عملية الحسابات في حياتنا اليومية بالنظام العشري ويتكون النظام العشري من عشرة أرقام هي ٠ و ١ و٢ و٣ و٤ و٥ و٦ و٧ و٨ و٩ ونقول ان اساس هذا النظام "١٠" لأنه يتألف من عشرة ارقام أما النظام العددي الذي يستخدم رقمين فقط هما ٠ و١ فيسمى بالنظام الثنائي واساسه يساوي "٢" .

والنظام الثنائي انتشر واشتهر مع انتشار الحاسوب لملاءمته لطبيعة الحاسوب البنائية .

ذكرنا أننا عندما نتحدث عن الرقم ٤٧٦٥ في النظام العشري فإننا نعني بأن

$$٤٧٦٥ = ٤ \times ١٠^{٣} + ٧ \times ١٠^{٢} + ٦ \times ١٠^{١} + ٥ \times ١٠^{٠}$$

$$= ٤ \times ١٠٠٠ + ٧ \times ١٠٠ + ٦ \times ١٠ + ٥ \times ١ = ٤٧٦٥$$

وبالمثل عندما نتحدث عن الرقم (٠١١٠) في النظام الثنائي فإننا نعني بان

$$(٠١١٠) = ٠ \times ٢^{٣} + ١ \times ٢^{٢} + ١ \times ٢^{١} + ٠ \times ٢^{٠}$$

$$= ٠ + ٤ + ٢ + ٠ = ٦$$

وهي تساوي ٦ بالنظام العشري .

٦. التحويل من النظام العشري إلى الثنائي وبالعكس :

للتحويل من النظام الثنائي للعشري نحسب ناتج أس ٢ ونضربها في معاملها (٠ أو ١) ثم نجمعها ، أما للتحويل من النظام العشري إلى النظام الثنائي يتم القسمة على ٢ ويحفظ خارج القسمة أسفل العدد والباقي . بمعنى العدد وتكرر هذه العملية إلى أن نصل إلى صفر ثم نقرأ عمود الباقي من أعلى إلى أسفل .

مثال (١٠) :

حول العدد الثنائي ١٠١٠٠٠١ الي عشري :

$$١٠١٠٠٠١ = ١ \times ٢^{٦} + ٠ \times ٢^{٥} + ١ \times ٢^{٤} + ٠ \times ٢^{٣} + ٠ \times ٢^{٢} + ٠ \times ٢^{١} + ١ \times ٢^{٠}$$

$$= ٦٤ + ٠ + ١٦ + ٠ + ٠ + ٠ + ١ = ٨١$$

إذن $(١٠١٠٠٠١)_{٢} = (٨١)_{١٠}$

معلومة

( ) تعني أن العدد ينتمي للنظام الثنائي $_{٢}$

( ) تعني أن العدد ينتمي للنظام العشري $_{١٠}$

مثال (١١) :

حول العدد الثنائي ١٠١٠١١١١ الي عشري :

$$(١٠١٠١١١١)_{٢} = ١\times٢^{٧} + ٠\times٢^{٦} + ١\times٢^{٥} + ٠\times٢^{٤} + ١\times٢^{٣} + ١\times٢^{٢} + ١\times٢^{١} + ١\times٢^{٠}$$

$$= ١ + ٢ + ٤ + ٨ + ٠ + ٣٢ + ٠ + ١٢٨ = ١٧٥$$

إذن $(١٠١٠١١١١)_{٢} = (١٧٥)_{١٠}$

مثال (١٢) :

حوّل ٢٥ إلى النظام الثنائي ثم مرة أخرى للعشري .

خارج القسمة	الباقي
٢٥	
١٢	١
٦	٠
٣	٠
١	١
٠	١

إذن $(٢٥)_{١٠} = (١١٠٠١)_{٢}$

العدد العشري	العدد الثنائي	التحويل إلى عشري مرة اخري
٢٥	$(١١٠٠١)_{٢}$	$= ١\times٢^{٠} + ٠\times٢^{١} + ٠\times٢^{٢} + ١\times٢^{٣} + ١\times٢^{٤}$
		$= ١ + ٠ + ٠ + ٨ + ١٦$
		$= ٢٥$

مثال (١٣) :

حول ٥٨ إلى النظام الثنائي ثم مرة أخرى إلى العشري

خارج القسمة	الباقي
٥٨	
٢٩	٠
١٤	١
٧	٠
٣	١
١	١
٠	١

العدد العشري العدد الثنائي التحويل إلى عشري

٥٨ (١١١٠١٠)٢ = ١×٢٥ + ٠×٢٤ + ١×٢٣ + ٠×٢٢ + ١×٢١ + ٠×٢٠ =

= ٠ + ٢ + ٠ + ٨ + ١٦ + ٣٢ = ٥٨

مثال (١٤) :

حول ١٠٧ إلى النظام الثنائي

خارج القسمة	الباقي
١٠٧	٢
٥٣	١
٢٦	١
١٣	٠
٦	١
٣	٠
١	١
٠	١

العدد العشري العدد الثنائي التحويل إلى عشري

١٠٧ (١١٠١٠١١) = ١×٢٠ + ١×٢١ + ٠×٢٢ + ١×٢٣ + ٠×٢٤ + ١×٢٥ + ١×٢٦

= ١ + ٢ + ٠ + ٨ + ٠ + ٣٢ + ٦٤

= ١٠٧

إذن (١٠٧)١٠ = (١١٠١٠١١)٢

٧. الكسور الثنائية :

إذا كان الكسر العشري ٠,٥٣٢ هو ١٠/٥ + ١٠٠/٣ + ١٠٠٠/٢ فإن (٠,١٠١)٢ هو ٢/١ + ٤/٠ + ٨/١ والذي يساوي ٠,٦٢٥ بالعشري، وإذا أردنا تحويل كسر عشري إلى كسر ثنائي فإننا نضرب الكسر في ٢ ونهمل العدد الصحيح ونكرر ضرب الكسر الناتج ، ونهمل العدد الصحيح إلى أن نصل إلى كسر كله أصفار كما في الأمثلة التالية :

مثال (١٥) :

حول الكسر العشري ٠,٣٢٢٤ إلى كسر ثنائي .

٠,٣٢٢٤
× ٢
٠,٦٤٤٨
× ٢
١,٢٨٩٦ يتم إهمال الواحد في الضرب
× ٢
٠,٥٧٩٢
× ٢
١,١٥٨٤ يتم إهمال الواحد في الضرب
× ٢
٠,٣١٦٨
× ٢
٠,٦٣٣٦
× ٢
٠,٢٦٧٢
× ٢
٠,٥٣٤٤
× ٢
٠,٠٦٨٨
× ٢
٠,١٣٧٦
× ٢
٠,٢٧٥٢
× ٢
٠,١٥٠٤
× ٢
٠,٣٠٠٨
× ٢
٠,٦٠١٦
× ٢

ونهمل العدد الصحيح ونكرر ضرب الكسر الناتج × ٢ ونهمل العدد الصحيح إلى أن يصبح الكسر كله أصفاراً .

مثال (١٦) :
حول الكسر العشري ٠,٢٥ إلى كسر ثنائي .

	٠,٢٥	
×	٢	
	٠,٥	٠
×	٢	
	٠٠	١

وعليه يصبح الكسر بالثنائي (٠,٠١) .
ولتحويله مرة أخرى إلى كسر عشري:

$$(٠,٠١)_{٢} = ٠ \times \frac{١}{٢} + ١ \times \frac{١}{٤} = \frac{١}{٤} = ٠,٢٥$$

إذن $(٠,٠١)_{٢} = (٠,٢٥)_{١٠}$

مثال (١٧) :
حول الكسر العشري ٠,٦٢٥ إلى كسر ثنائي .

	٠,٦٢٥	
×	٢	
	٠,٢٥٠	١
×	٢	
	٠,٥٠٠	٠
×	٢	
	٠٠	١

وعليه يصبح الكسر بالثنائي $(٠,١٠١)_{٢}$ ولتحويله مرة أخرى إلى عشري :

$$(٠,١٠١)_{٢} = ١ \times \frac{١}{٢} + ٠ \times \frac{١}{٤} + ١ \times \frac{١}{٨} = \frac{٥}{٨} = ٠,٦٢٥$$

إذن $(٠,١٠١)_{٢} = (٠,٦٢٥)_{١٠}$

مثال (١٨) :

حول الكسر $٥/٧$ إلى كسر ثنائي .

أولاً يتم تحويل الكسر إلى كسر عشري :

$\frac{٥}{٧}$ = ٠,٧١٤٢٨٥٧١٤٢٨٥ (كسر دائري)

بالتقريب إلى ستة خانات عشرية

	٠,٧١٤٢٨٦	
×	٢	
	٠,٤٢٨٥٧٢	١
×	٢	
	٠,٨٥٧١٤٤	٠
×	٢	
	٠,٧١٤٢٨٨	١
×	٢	
	٠,٤٢٨٥٧٦	١
×	٢	
	٠,٨٥٧١٥٢	٠
×	٢	
	٠,٧١٤٣٠٤	١
×	٢	
	٠,٤٣٢٦٠٨	١

يمكن الاكتفاء بهذه الدقة وتقريب التحويل الثنائي (٠,١٠١١٠١١)٢ أي لسبع خانات ثنائية .

مثال (١٩) :

حول $٥\frac{١}{٢}$ إلى كسر ثنائي .

يتم أولاً تحويل العدد الصحيح إلى ثنائي وهو :

٥ = (١٠١)٢ ، ثم تحويل ٥. إلى كسر ثنائي وهي ٠,١ ويمكن أن يصبح التمثيل الثنائي لـ $٥\frac{١}{٢}$ هـو (١٠١,١)٢ .

٨. الجمع الثنائي :

يتم الجمع الثنائي بنفس فكرة الجمع العشري، أي إذا كان المجموع أكثر من واحد أي اثنين أو ثلاثة فإننا نضع في خانة الجمع صفراً أو واحداً ويحمل واحد إلى الخانة التي على اليسار وذلك حسب جدول الجمع التالي :

١	٠	+
١	٠	٠
٠*	١	١

"*" تعني ان يتم حمل "١" الي الخانة التالية (في النظام العشري نجد ان ٩+١=٠ ومعنا "١" اي ان نحمل "١" الي الخانة التالية)

مثال (٢٠) :

١
+ ١
+ ١
= ١ ١

وذلك لأن

١
+ ١
= ١ ٠
+ ١
= ١ ١

مثال (٢١) :

١
+ ١
+ ١
+ ١
= ١ ٠ ٠

وذلك لأن

١
+ ١
= ١ ٠

وان ١

+ ١
= ١ ٠
+ ١ ٠
= ١ ٠ ٠

مثال (٢٢) :

١ ١ ١ ١ ٠ ٠ ١ ١ ١ ١
+ ١ ٠ ١ ١ ٠ ١ ١ ٠ ١
= ١ ٠ ١ ٠ ١ ١ ١ ١ ٠ ٠

مثال (٢٣) :

	٩	=	١ ١ ١
+	٧	=	١ ٠ ٠ ١
=	١٦		١ ٠ ٠ ٠ ٠

إذن $(١٦)_{١٠} = (١٠٠٠٠)_{٢} = ٠\times٢^{٠} + ٠\times٢^{١} + ٠\times٢^{٢} + ٠\times٢^{٣} + ١\times٢^{٤}$

$= ٠ + ٠ + ٠ + ١٦ = ١٦$

مثال (٢٤) :

اجمع ٣٨ + ٥٩ + ٨٨

نجمع أولاً ٣٨ + ٥٩

	٣٨	=	١ ٠ ٠ ١ ١ ٠
+	٥٩	=	١ ١ ١ ٠ ١ ١
=	٩٧	=	١ ١ ٠ ٠ ٠ ٠ ١
+	٨٨	=	١ ٠ ١ ١ ٠ ٠ ٠
	١٨٥	=	١ ٠ ١ ١ ١ ٠ ٠ ١

إذن $(١٨٥)_{١٠} = (١٠١١١٠٠١)_{٢}$

$= ١\times٢^{٠} + ٠\times٢^{١} + ٠\times٢^{٢} + ١\times٢^{٣} + ١\times٢^{٤} + ٠\times٢^{٥} + ٠\times٢^{٦} + ١\times٢^{٧}$

$= ١ + ٠ + ٠ + ٨ + ١٦ + ٣٢ + ٠ + ١٢٨ = ١٨٥$

٩. الطرح الثنائي :

كذلك يتم الطرح الثنائي مثل الطرح العشري ويتم الاستلاف من الخانة اليسرى في حالة طرح واحد من صفر ونراعي عند الاستلاف ان٢ المستلفة في النظام الثنائي تماثل استلاف العشره في النظام العشري من خانة الي اخري .

١	٠	–
١	٠	٠
	١٠*	١

"*" تعني أن يتم استلاف "١" من الخانة التالية (في النظام العشري نجد ان ٠-٩=١ ونطرح "١" من الخانة التالية) .

مثال (٢٥) :

احسب ٣٧ – ٢١ بالنظام الثنائي .

	٣٧ =	١٠٠١٠١
+	٢١ =	١٠١٠١
=	١٦	٠١٠٠٠٠

مثال (٢٦) :

احسب ٣٧ – ٣١ بالنظام الثنائي

	٣٧ =	١٠٠١٠١
+	٣١ =	١١١١١
=	٦	٠٠٠١١٠

مثال (٢٧) :

احسب ٨/٣ ١١ – ١٦/١١ ٦ بالنظام الثنائي

٨/٣ ١١ = ١ ٠ ١ ١ , ٠ ١ ١ ٠

+ ١٦/١١ ٦ = ١ ١ ٠ , ١ ٠ ١ ١

= ١٦/١١ ٤ = ٠ ١ ٠ ٠ , ١ ٠ ١ ١

إذن(١٦/١١ ٤)١٠ = (١٠٠,١٠١١)٢

١٠. الضرب الثنائي والقسمة الثنائية :

الضرب كما نعلم هو من حيث المبدأ عملية جمع متكرر أي ٥ × ٦ هو ٦+٦+٦+٦+٦ وكذلك القسمة هي من حيث المبدأ عملية طرح متكررة فقسمة ٧ على ٢ مثلاً هي :

[١] ٧ – (١ + ١) = ٥ ، ثم

[٢] ٥ – (١ + ١) = ٣ ، ثم

[٣] ٣ – (١ + ١) = ١ ، ثم

[٤] ١ – (١ + ١) لاتكفي للطرح ويمكن أن تكون النتيجة ٣ والباقي ١ . نلاحظ أن وضعنا داخل القوس ١ + ١ أي نعطي كل واحد من المقسوم عليهم سهماً في كل مرة . .

إن أهمية فهم الضرب والقسمة كعمليتي جمع وطرح تأتي عند تقدير وقت الحاسوب في العمليات الحسابية، فوقت الضرب والقسمة هو أضعاف وقت الجمع والطرح؛ لأنه وقت عمليات جمع وطرح مكررة وكذلك وبنفس المبدإ وقت عملية حساب الأس هو أضعاف وقت الضرب والقسمة لأن الأس ناتج عمليات ضرب أو قسمة مكررة .

إن عملية الضرب والقسمة الثنائية تتم بنفس طريقة الضرب والقسمة العشرية على نحو الأمثلة التالية :

مثال (٢٨) :

احسب ٩× ١١ بالنظام الثنائي

٩	=	١ ٠ ٠ ١
× ١١	=	١ ٠ ١ ١
=	=	١ ٠ ٠ ١ ٠ ٠ ٠
		٠ ٠ ٠ ٠ ٠ ٠
		١ ٠ ٠ ١ ٠
		١ ٠ ٠ ١
٩٩		١ ١ ٠ ٠ ٠ ١ ١

مثال (٢٩) :

احسب ٣١ × ٢٧ بالنظام الثنائي

٣١	=	١ ١ ١ ١ ١
× ٢٧	=	١ ١ ٠ ١ ١
=	=	١ ١ ١ ١ ١ ٠ ٠ ٠ ٠
		١ ١ ١ ١ ١ ٠ ٠ ٠
		١ ١ ١ ١ ١ ٠
		١ ١ ١ ١ ١
٨٣٧		١ ١ ٠ ١ ٠ ٠ ٠ ١ ٠ ١

لإكمال عملية الجمع يتم جمع الصفين الأولين ثم يضاف للناتج الصف الثالث، ثم يضاف للناتج الصف الرابع، وهكذا.

مثال (٣٠) :

اقسم ١٠ على ٥ ثنائياً : ١٠ = $(١٠١٠)_{٢}$ ÷ ٥ = $(١٠١)_{٢}$

```
        ١ ٠
    ١ ٠ ١ | ١ ٠ ١ ٠
          -----
          ١ ٠ ١
          -----
            ٠ ٠
              ٠
            ---
            ٠
```

إذن $(١٠١٠)_{٢}$ ÷ $(١٠١)_{٢}$ = $(١٠)_{٢}$ = $(٢)_{١٠}$

مثال (٣١):

اقسم ٤٢ على ٧ ثنائياً : $(١٠١٠١٠)_{٢}$ ÷ $(١١١)_{٢}$

```
          ١ ١ ٠
  ١ ١ ١ | ١ ٠ ١ ٠ ١ ٠
        -------------
          ٠ ١ ١ ١
          -------
            ١ ١ ١
            ١ ١ ١
            -----
              ٠ ٠
              ===
```

إذن $(١٠١٠١٠)_{٢}$ ÷ $(١١١)_{٢}$ = $(١١٠)_{٢}$ = $(٦)_{١٠}$

مثال (٣٢) :

اقسم ٢٤٩ على ٤٢ ثنائياً
أي $(١٠٠١٠٠١١٠)_{٢} \div (١٠١٠١٠)_{٢}$

١ ١ ١
١٠١٠١٠ | ١٠٠١٠٠١١٠
٠١٠١٠١٠
٠١١١١١١٠
٠١٠١٠١٠
١٠١٠١٠
١٠١٠١٠
٠٠٠٠٠٠
٠
٠

إذن $(١٠٠١٠٠١١٠)_{٢} \div (١٠١٠١٠)_{٢} = (١١١)_{٢} = (٧)_{١٠}$

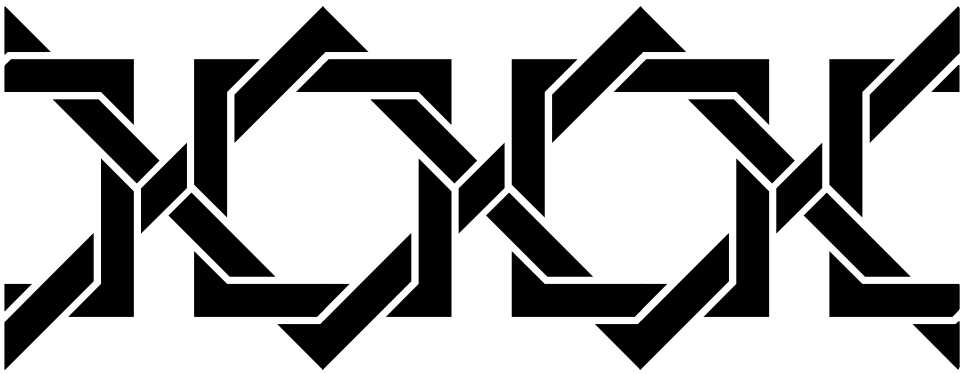
مثال (٣٣) :

اقسم $٩\frac{١}{٢}$ على $٤\frac{٣}{٤}$ ثنائياً
أي $(١٠٠١,١)_{٢} \div (١٠٠,١١)_{٢}$

$= (100110)_2 \div (10011)_2$

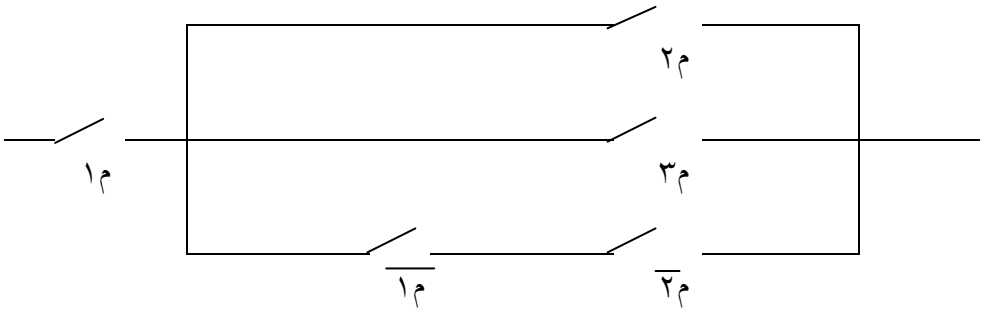
$$
\begin{array}{r|l}
100110 & 10011 \\
10011 & 10 \\
\hline
00 & \\
0 & \\
\hline
0 &
\end{array}
$$

إذن $(100110)_2 \div (10011)_2 = (10)_2 = (2)_{10}$

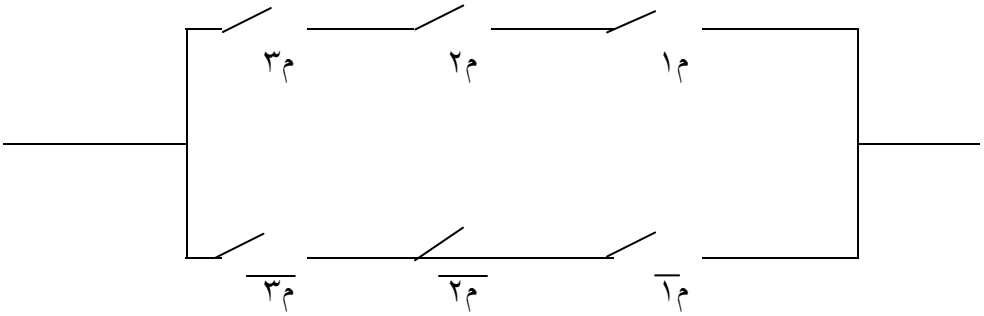


١١. تمرين

١- ما ناتج الدائره الاتيه ؟



٢- عبر بالجبر المنطقى عن الدائرة التالية :



٣- كيف يعالج صمام NOT كل متتابعه من الآتى ؟

أ ١ ١ ١ ٠ ٠ ٠ ١ ٠ ١

ب ١ ٠ ١ ١ ١ ١ ٠ ٠ ١ ٠ ١

ج ١ ١ ٠ ٠ ١ ١ ١ ١ ١ ١ ١

د ١ ٠ ٠ ٠ ٠ ٠ ٠ ٠ ٠ ٠ ٠ ١

٤- أكمل الجدول التالى:

١ ١ ١ ١ ٠	١ ١ ١ ١ ٠	الصمام
.....	.....	١ ١ ٠ ٠ ١
.....	.....	١ ١ ١ ١ ٠
.....	.....	١ ٠ ٠ ١ ١

إذا كان الصمام :-

(أ) OR (ب) AND

٥- إذا كان :

أ ١ ١ ١ ١ ٠ ٠ ١ ٠ ١ ٠ ٠ ٠

ب ١ ٠ ١ ١ ١ ١ ٠ ٠ ١ ٠ ٠ ١

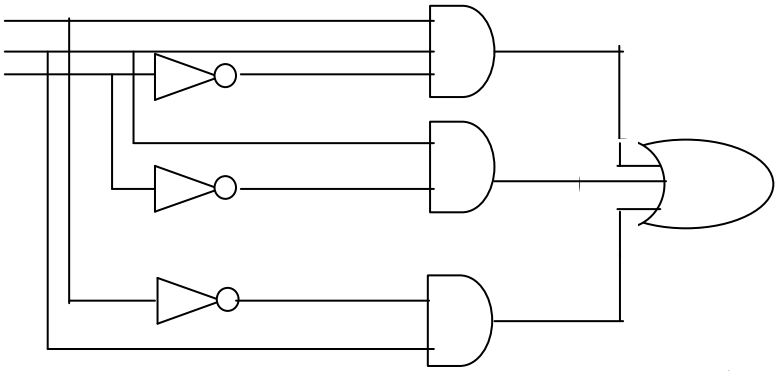
ج ١ ٠ ٠ ٠ ١ ١ ٠ ١ ١ ٠ ٠ ١

جد :

(أ) أ+ب+ج (ب) أ×ب×ج

(ج) أ×(ب+ج) (د) ب×ج×أ

٦- اختصر الدائرة الكهربية التالية، ثم جد جدول الصواب لها :



٧- إذا كانت :

أ ١ ٠ ١ ١ ٠ ١ ٠ ١ ١

ب ١ ٠ ١ ١ ٠ ١ ١ ٠ ١

ج = أ ⊕ ب

جد قيم ج "ثنائي" إذا كانت العلامة ⊕ تعنى :

i. جمع
ii. قسمة
iii. طرح
iv. ضرب

٨- إذا كانت :

أ = 9½

ب = 4¾

ج = أ ⊕ ب

جد قيم ج "ثنائي" إذا كانت العلامة ⊕ تعنى :

i. جمع
ii. قسمة
iii. طرح
iv. ضرب

٩-حول العدد الثنائي التالي الى عشري :

i. ١٠١٠١١١١

ii. ١١١١١١١١

iii. ١٠١٠١١٠١١٠

iv. ١١١١١١١١٠١

v. ١٠١٠١٠١٠١٠٠

vi. ١١١٠١

vii. ١١١١٠٠٠١١١

viii. ١٠٠١٠٠١٠٠

ix. ١٠٠٠

١٠ – حول العدد العشري التالي الى ثنائي :

i. ١٥٧

ii. ٩٤ ١/٢

iii. ١٤٣

iv. ١٩ ٣/٤

v. ٤٣٢

vi. ٢٣٢

vii. ١٧

viii. ١١١

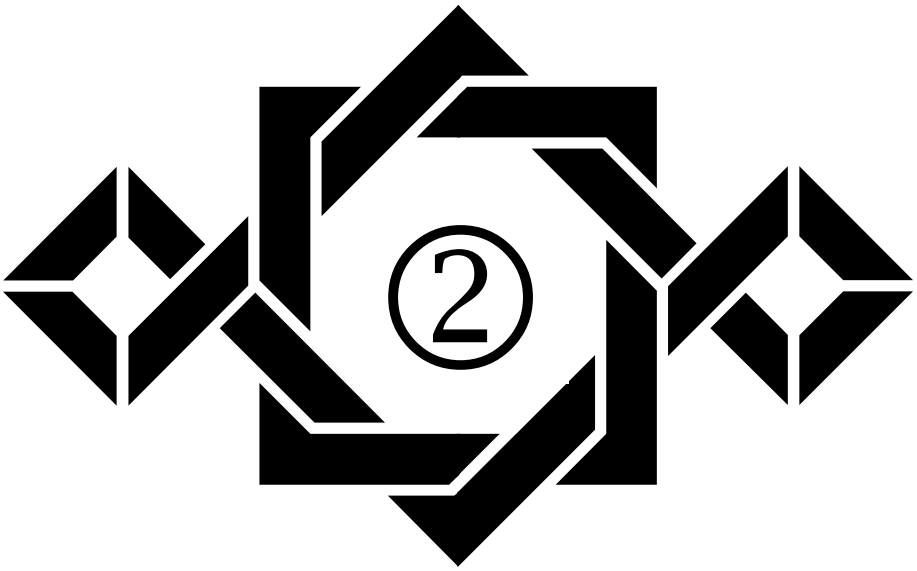
ix. ١٠٠٠

x. ١٠٩

xi. ١٠٨

xii. ١١١

xiii. ١٢١



بنائيّات البيانات

بنائيات البيانات

١. مدخل :

ذكرنا في كتاب الصف الثاني أن نظام معالجة البيانات المحوسب يعني استخدام الحاسوب في تخزين البيانات ثم استرجاعها بعد عمل المعالجات المطلوبة عليها ، وذكرنا أن المعالجة ربما تعني البحث عن معلومة معينة ،أو تصنيف البيانات تصنيفاً معيناً ، أو تحليلها وفق نموذج رياضي معين، أو رسمها رسماً بيانياً معيناً، أو تأمينها وإخفاءها وفق شفرة سرية معينة، أو غير ذلك من المعالجات المختلفة . وحتى نتمكن من .فعل كل ذلك بكفاءة عالية وفي أقل وقت ممكن وفي أقل مساحة تخزينية ممكنة لابد أن نتعرف علماً مهماً من علوم الحاسوب يسمى بنائيات البيانات (data structures) ونعني ببنائيات البيانات الخوارزميات(algorithms) التي تمكن من تنظيم وتخزين البيانات (المعالجة) بالصورة المطلوبة في أسرع وقت ممكن وفي أقل سعة تخزينية ممكنة .

٢. ذاكرة الحاسوب :

تتكون ذاكرة الحاسوب من عدد كبير من مواقع تخزين البيانات وهذه المواقع متساوية أي أن كل موقع يتكون من عدد ثابت من الثنائيات أو البتات(Bits) يسمى كلمة (word) وهذا العدد يختلف من حاسوب لآخر حسب حجم وقوة الحاسوب فعلى سبيل المثال هناك ٨ و١٦ و٣٢ و٦٤ و١٢٨ ثنائية للكلمة . والكلمة الواحدة يمكن أن تقسم إلى ثمانيات أو بايتات (Bytes) حيث كل ثمانية أو بايت تمثل رمزاً كما ذكرنا في منهج الصف الأول . لكل كلمة من كلمات الذاكرة عنوان خاص بها يمكن عن طريقه الوصول إليها ، أما في حالة الكلمات طويلة الثنائيات فهناك عنوان داخل عنوان الكلمة يوصل إلى الرمز المطلوب (البايت) داخل تلك الكلمة .

تصنف البيانات المخزنة في ذاكرة الحاسوب إلى :

- بيانات لا يمكن تغيرها أو بعبارة أخرى أن مواقعها لا تتقبل بيانات جديدة وتسمى بمواقع أو عناوين الذاكرة الثابتة .
- مواقع أو عناوين الذاكرة المتغيرة أي تلك التي تتقبل تغيير في بياناتها حسب توجيهات البرنامج ومثال لذلك رصيد العميل في البنك والذي يتغير كلما سحب العميل مالاً أو أودع مالاً .

٣. أنواع البيانات :

تتفق كل لغات البرمجة في الحد الأدنى على أربعة أنواع من البيانات هي:

- الأعداد الرقمية (Integers) أي الأعداد الصحيحه مثل ١، ٥،-٢٢٥ ..
- الأعداد الحقيقية أي الأعداد التي بها خانة عشرية مثل .٠٥ ، ٣٦,٠٥ ..
- البيانات الحرفية character ونعني بها كل مفاتيح الطباعة من حروف وعلامات وأرقام (مخزنة كأنها حروف وليست أعداد)
- البيانات المنطقية (logical) أو (Boolean) وهي التي تأخذ فقط قيمتي خطأ أو صواب .

لما كانت ذاكرة الحاسوب لا تستقبل أو تتعامل مع البيانات إلا في صورة ثنائية – كان لابد من مراعاة هذه الأنواع الأربعة للبيانات عند الترميز ويتم ذلك عادة عن طريق جدول خاص يسمى جدول الترميز يحدد نوع البيانات في كل عنوان من عناوين الذاكرة إن كان به بيانات .

يسمى عنوان الذاكرة سواء أكان ثابتاً أم متغيراً **عنواناً بسيطاً** إذا كان يشير إلى موقع واحد فقط بالذاكرة إذن العنوان يمكن أن يكون مركباً من عدة مواقع في الذاكرة كما سنرى لاحقاً .

٤. ترميز الأعداد الرقمية :

هناك ثلاث خوارزميات مشهورة في ترميز الأعداد الرقمية ثنائياً ، كل هذه الخوارزميات تتفق في أن الثنائية الأولى في أقصى اليسار (أقصى اليمين في

الكتابة العربية) تكون لتحديد علامة الرقم أي علامة السالب والموجب، وعادة يعطي الصفر لعلامة الموجب، والواحد لعلامة السالب . كذلك تتفق في تمثيل قيمة الرقم الموجب بالتمثيل العددي الثنائي العادي مثلاً :

تمثل + ٢ بـــ : $2_{10} = 00\ldots\ldots\ldots\ldots 00010_2$

وتمثل + ٧ بـــ : $7_{10} = 00\ldots\ldots\ldots\ldots 00111_2$

وتمثل + ٩ بـــ : $9_{10} = 00\ldots\ldots\ldots\ldots 01001_2$

ويمثل + صفر بـــ : $0_{10} = 00\ldots\ldots\ldots\ldots 00000_2$

أما الأرقام السالبة فتختلف من خوارزمية لأخرى :

الخوارزمية الأولى :

تعرف هذه الخوارزمية بخوارزمية علامة القيمة ، فهي تتعامل مع قيم الأرقام السالبة مثل قيم الأرقام الموجبة مع إعطاء بنائية العلامة القيمة واحد بالنسبة للأعداد السالبة فمثلاً :

تمثل – ٢ بـــ : $-2_{10} = 100\ldots\ldots\ldots\ldots 00010_2$

وتمثل – ٧ بـــ : $-7_{10} = 100\ldots\ldots\ldots\ldots 00111_2$

وتمثل – ٩ بـــ : $-9_{10} = 100\ldots\ldots\ldots\ldots 01001_2$

ويمثل – صفر بـــ : $-0_{10} = 100\ldots\ldots\ldots\ldots 00000_2$

نلاحظ هنا أن هذه الخوارزمية أعطت موجب صفر وسالب صفر تمثيلين مختلفين، وبهذا سيفيد الحاسوب حسب هذه الخوارزمية أنهما غير متساويين مع أنهما في الواقع يتساويان، لهذا لابد أن يفهم الحاسوب بطريقة أخرى بأنهما متساويان وهذا يعتبر من عيوب هذه الخوارزمية .

الخوارزمية الثانية :

وتعرف بخوارزمية الاتمام (الإكمال) الآحادي وهي تمثل قيمة الرقم السالب بعكس ثنائيات الرقم الموجب، أي أن أي ثنائية قيمتها صفر تصبح قيمتها واحداً وأي ثنائية قيمتها واحد تصبح قيمتها صفراً على النحو التالي :

تمثل – ٢ بـــ : $-2_{10} = 11\ldots\ldots\ \ldots\ldots\ 11101_2$

وتمثل – ٧ بـــ : $-7_{10} = 11\ldots\ldots\ldots\ldots\ 11000_2$

وتمثل – ٩ بـــ : $-9_{10} = 11\ldots\ldots\ldots\ldots\ 10110_2$

وتمثل – ٠ بـــ : $-0_{10} = 11\ldots\ldots\ldots\ldots\ 11111_2$

هذه الطريقة لها نفس عيوب الطريقة الأولى في إعطاء الصفر تمثيلين مختلفين ولكن لها ميزة على الطريقة الأولى إذ أن الجمع والطرح يتم بطريقة مباشرة دون تعقيد، ودون النظر أو إعطاء اعتبار لثنائية العلامة بل يتم التعامل فيها كجزء من الرقم وإذا كان هناك واحد زائد بعد العلامة يضاف في أقصى اليمين مرة أخرى .

الخوارزمية الثالثة :

وتعرف بخوارزمية الاتمام (الإكمال) الثنائي وهي تتم بنفس المعاملة التي تمت في الخوارزمية الثانية ويضاف واحد للرقم أي بجمع واحد في أقصى اليمين مع تمثيل الإكمال الأحادي .

تصبح – ٢ $-2_{10} = 11\ldots\ldots\ldots\ldots\ 11110_2$

وتصبح – ٧ $-7_{10} = 11\ldots\ldots\ldots\ldots\ 11001_2$

وتصبح – ٩ $-9_{10} = 11\ldots\ldots\ \ldots\ldots\ 10111_2$

وتصبح – صفر $-0_{10} = 00\ldots\ldots\ldots\ldots\ 00000_2$

هنا نلاحظ أن سالب صفر وموجب صفر يأخذ نفس القيمة، كما نلاحظ أن الجمع والطرح يتم بطريقة عادية ودون أي اعتبار لثنائية العلامة أو الواحد الزائد بعد ثنائية العلامة .

مثال (١) :

اجمع موجب ٩ مع سالب ٧ مستخدماً الخوارزميات الثلاث في تمثيل الأرقام :

أ) خوارزمية العلامة والقيمة :

$+9_2$ = 0 0 0 0 0 0 1 0 0 1
-7_2 = 1 0 0 0 0 0 0 1 1 1
2_2= 1 0 0 0 0 1 0 0 0 0

هذه الطريقة لا تعطي الجواب الصحيح "2" إذا عملنا جمعاً مباشراً ولكن هناك معالجات أخرى لتصحيح الخطأ .(ليس من اهداف هذا المنهج)

ب) خوارزمية الاتمام الآحادي :

9_2 = 0 0 0 0 - - 0 1 0 0 1
-7_2 = 1 1 1 1 - - 1 1 0 0 0

0 0 0 0 - - 0 0 0 0 1
1
2 = 0 0 0 0 - - 0 0 0 1 0

ينقل الواحد من الخانة الزائده الي اقصي اليمين ثم يجمع مرة اخري ليعطي النتيجة الصحيحة "٢"

ج) خوارزمية الاتمام الثنائي:

9 = 0 0 - - - - - - 0 1 0 0 1
-7 = 1 1 - - - - - - 1 1 0 0 1
2 = 0 0 - - - - - - 0 0 0 1 0

حيث يتم اعتبار ان الواحد في الخانة الزائدة لا شئ وكانت النتيجه صحيحة "٢"

٥. تمثيل أنوع البيانات

أولاً: الأعداد الحقيقية :

تخزن الأعداد الحقيقية في جزئين من الكلمة ، الجزء الأول يخزن فيه الكسر ويسمى المانتيسا (mantissa) أو الخانات الكسرية المؤثرة . أما الجـزء الثـاني فيخزن فيه القوة أو الأساس، والعدد الحقيقي عبارة عـن ضـرب الجـزء الأول

(الكسري) في الأساس مرفوعاً للقوة التي بالجزء الثاني . وفي الجزء الكسري لابد أن يكون الرقم المجاور للخانة العشرية أكبر من صفر، أي لا يقبل الصفر بأن يكون مجاوراً للخانة العشرية مباشرة ، وهذا الشرط يسمى تطبيع الكسر فمثلاً المليون يمكن أن يكتب ١ × ١٠^٦ أو ١٠٠ × ١٠^٤ ولكن في التمثيل الطبيعي بالحاسوب فإنه يكتب ٠,١ × ١٠^٧ ويخزن ٠,١ في الجزء الكسري (المانتيسا) وتخزن ٧ في الجزء الثاني جزء القوة. كذلك ٠,٠٠٢٥ تأخذ الشكل ٠,٢٥ × ١٠^-٢ ويخزن ٠,٢٥ في الجزء الكسري وتخزن -٢ في جزء القوة ومثلها ٣٠٠٠ فإنها تصبح ٠,٣ × ١٠^٤ وتخزن ٠,٣ في الجزء الكسري و٤ في جزء القوة .

على سبيل المثال إذا كان لدينا حاسوب طول كلمته ٤٠ ثنائية يمكن أن نقسم الكلمة لتخزين العدد الحقيقي على النحو التالي :

الكسر	علامة الكسر	القوه	علامة القوه
٣٢ ثنائية	١ ثنائية	٦ ثنائية	١ ثنائية

العمليات الحسابية في الأعداد الحقيقية ليست ببساطة العمليات الحسابية مع الأرقام فمثلاً خوارزمية الجمع تتم على النحو التالي :

أ. حرّك العلامة العشرية (أو الثنائية) للعدد الأصغر نحو الشمال حيث تصبح قوة العدد الأصغر مثل قوة العدد الأكبر .

ب. اجمع كسري العددين فقط .

ج. طبع الكسر الناتج بإزالة الأصفار وتعديل القوة على ضوء ذلك ثم قرب الكسر حسب المساحة المتاحة للجزء الكسري .

مثال (٢) :

اجري العملية ٠,٠٠٢٥ + ٣٠٠٠ علي حاسوب يتيح ٦ خانات في الجزء الكسري

الوضع الطبيعي : ٠,٢٥ × ١٠$^{-٢}$ + ٠,٣ × ١٠^{٤}

- حرك علامة العدد الأصغر حتى يأخذ نفس قوة العدد الأكبر

  ٠,٢٥ × ١٠$^{-٢}$ = ٠,٠٠٠٠٠٠٢٥ × ١٠^{٤}

- اجمع ٠,٣٠٠٠ + ٠,٠٠٠٠٠٠٢٥ = ٠,٣٠٠٠٠٠٢٥ × ١٠^{٤}
- طبع الناتج الي سته خانات أي الناتج = ٠,٣٠٠٠٠٠ × ١٠^{٤} إذا كان الحاسوب يتيح ستة خانات فقط في الجزء الكسري .

ثانياً الحروف :

يتم تمثيل الحروف حسب نظم الترميز المعروفة والتي تم وصفها في كتاب الصف الأول مثل نظام آسكي (ASCII) ونظام ابسدك (EBCDIC)، والنظام الأول هو النظام القياسي ، أما النظام الثاني فهو نظام آي بي أم (IBM) وهي أكبر شركات الحواسيب في العالم، ويختلف عن النظام الأول في أنه يبدأ بترميز الحروف الصغيرة ثم الحروف الكبيرة، ثم الأرقام. أما النظام الأول فيقوم بترميز الأرقام ثم الحروف الكبيرة ثم الحروف الصغيرة .

ثالثاً : البيانات المنطقيه :

يتم تمثيل البيانات المنطقية بثنائية واحدة فقط عندما تكون قيمتها واحداً تعنى صحيح أو حقيقية وعندما تكون قيمتها صفراً تعني خطأ أو كذباً .

٦. التحول في نوع البيانات :

تتيح لغات البرمجة التحول من نوع بيانات لآخر، فمثلاً يمكن أن نحوّل الأعداد الرقمية إلى أعداد حقيقية بسهولة حيث فقط تضاف لها العلامة العشرية ثم تطبع مثلاً الرقم ٢٨ يصبح ٠,٢٨ × ١٠^{٢} وهكذا . أما التحول من عدد حقيقي إلى عدد رقمي فيتم إما بالتقريب أو بالكشط وكلتا الطريقتين متاحتان في لغات البرمجة فمثلاً ٣٢٥,٧ و التي تمثل ب ٠,٣٢٥٧ × ١٠^{٣} يمكن أن تصبح بالتقريب العدد الرقمي ٣٢٦ أو بالكشط العدد الرقمي ٣٢٥ . أما التحول من حرفي إلى رقمي فيتم

بإعطاء الحرف قيمة العدد في التمثيل. مثلاً إذا كان الحرف أ ممثل بـــ ١٠٠٠٠٠٠٠١ فإن تحويله إلى عدد عشري تصبح ٢٥٧ أي ١ × ٢٠ + ٢×٠١ + ٢×٠٢ + ٢×٠٣ + ٢×٠٤ + ٢×٠٥ + ٢×٠٦ + ٢×٠٧ + ٢×٠٨ =١+٢٥٦=٢٥٧.

بعض لغات البرمجة مثل بيسك وفورتران لا تشترط تعريف نوع البيانات في المتغيرات، وهذه بالتأكيد لها بعض المشاكل، ولكن لغة فورتران تعرف تلقائياً نوع المتغير حسب الحرف الأول المستخدم، فإذا كان الحرف الأول بين I و N فإنها تعتبر المتغير عدداً رقمياً وتعتبر الحروف غير ذلك متغيراً عددياً حقيقياً . أما اللغات الأخرى مثل باسكال، و سي وهي أهم اللغات وأكثرها انتشاراً في الوقت الحالي فإنها تقوم بتعريف نوع المتغيرات في أول البرنامج .

٧. المصفوفات والسجلات :

إن المتغير البياني البسيط أو الذي يعبر عن وحدة بيانية مفردة لا يحقق كل الأغراض، فهناك متغيرات كثيرة تحتاج في وصفها لمجموعة وحدات بيانية. فمثلاً إذا كنا نريد أن نتحدث عن الطالب فإننا نتحدث عن اسمه، وتاريخ ميلاده، وقريته، والسنة الدراسية التي يدرس فيها، وعن ترتيبه، وعن درجاته، وعن سلوكه .. الخ فكل هذه الوحدات البيانية تتفق أو تشترك في وصف الطالب . إذن هناك متغير أساسي هو الطالب مركب من متغيرات مفردة وهي أوصاف الطالب، وكل مفردة من هذه المفردات تحتاج إلى موقع في الذاكرة . كل مواقع الذاكرة التي ترتبط بوصف الطالب تكون بنائية مركبة، وأكثرها شيوعاً واستخداماً هي بنائية المصفوفات وبنائية السجلات . المصفوفة عبارة عن بنائية بيانية مركبة لها المواصفات التالية :

- كل وحداتها البنائية من نوع واحد أي كل بيانات المصفوفة يجب أن تكون أعداداً حقيقية أو أعداداً رقمية أو حروفاً أو منطقية .

- تتكون المصفوفة من أبعاد مركبة وكل بعد مركب من عدد من الوحدات البنائية ليس بالضرورة أن يكون مساوياً للأبعاد الأخرى وبالطبع أبسط أنواع المصفوفات المصفوفة ذات البعد الواحد أي تلك التي بها عمود واحد فقط مثلاً أسماء طلاب الفصل يمكن أن تخزن في مصفوفة ذات بعد واحد .

مثل :

أسماء الطلاب

١- ابوذز عثمان

٢- خالد غازي

٣- محمد الطيب عقال

٤- وفاء قمر

٥- صالح عوض

٦- عبدالله النقيب

.

.

.

٤٩- محمد قمر

٥٠- الجيلي أنس

هذه المصفوفة لها بعد واحد ونوعها حرفية وطول البعد = ٥٠

يمكن أن نصمم مصفوفة أكثر تعقيداً تحتوي علي ترتيب الطالب، ومجموع الطالب، ودرجة الطالب في المواد المختلفة وهي: اللغة العربية واللغة الإنجليزية والدراسات الإسلامية والرياضيات والفيزياء والكيمياء والأحياء على النحو أدناه :

هذه المصفوفة رقمية ولها بعدان بعد طوله ٥٠ والآخر طوله تسعة ويمكننا تعقيد هذه المصفوفة بإضافة بعد ثالث طوله خمسة عبارة عن درجات الطالب في الأسئلة الخمسة في كل مادة .

١	٦٥٠	٩٠	٩٠	٩٥	٩٥	٩٥	٩٥	٩٠
٤	٦٤٢	٩٢	٩٠	٩٠	٩٥	٩٥	٩٠	٩٠
٣	٦٤٥	٩٠	٩٥	٩٠	٩٥	٩٥	٩٠	٩٠
٢	٦٤٧	٩٠	٩٥	٩٥	٩٥	٩٢	٩٠	٩٠
.								
.								
٥٠	٢٠٠	٤٠	٣٠	٥٠	٢٠	٢٠	٢٠	٢٠

فإذا اسمينا المصفوفة الأولى التي تحتوي علي الأسماء فقط ب(أسماء الطلاب) فإننا عندما نسأل عن أسماء الطلاب (٣) فإننا نسأل عن اسم الطالب الذي في الموقع الثالث في المصفوفة، وهو محمد الطيب عقال. أما المصفوفة الثانية والتي أسميناها درجات الطلاب فإنه عندما يُسأل عن درجات الطلاب (٢،٣) نسأل عن البيان الذي يقع عند تقاطع الصف الثاني مع العمود الثالث وهي تساوي "٩٢" عبارة عن درجة الطالب في اللغة االعربية حسب الترتيب الذي صممنا على ضوئه المصفوفة وهي أن يكون العمود الأول للترتيب والعمود الثاني للمجموع والعمود الثالث للغة العربية أما إذا سألنا درجات الطلاب (٢ ، ١) فإن الحاسوب سيجيب ٤ أي أن البيان الذي يقع في تقاطع الصف الثاني مع العمود الأول وهو ترتيب الطالب الثاني في قائمة الأسماء وترتيبه الرابع. لاحظ أن هناك فرقاً بين ترتيب الطالب في المجموع، وترتيبه حسب قائمة الأسماء . أما إذا نظرنا إلى المصفوفة الأكثر تعقيداً

أي تلك التي بها درجات الأسئلة الخمس في كل مادة وسألنا عن المعلومة تفاصيل الطالب (٣ ، ٤ ، ٥) – فإننا نسأله عن درجة الطالب في السؤال الخامس في العمود الرابع، وهو مادة اللغة الانجليزية والطالب المعني هو رقم ٣ في قائمة أسماء الطلاب .

يتم تخزين المصفوفة في الحاسوب على النحو التالي :

أ تخزين الموقع الأساس (base location) وهو الموقع بالذاكرة الذي يبدأ منه على التوالي تخزين عناصر المصفوفة .

ب تخزين عدد أبعاد المصفوفة، وطول البعد أي عدد المتغيرات في كل بعد في المثال السابق هناك ثلاث أبعاد :البعد الأول وطوله ٥٠ (للطلاب)، والبعد الثاني وطوله ٩(للمواد)، والبعد الثالث وطوله ٥(اسئله في كل مادة).

ج تخزين أرقام المؤشرات لكل عنصر في المصفوفة، مثلاً في المصفوفة الأولى المؤشر رقم (٢) يشير إلى خالد غازي مثلاً . أما المصفوفة الثانية فلها مؤشران: مؤشر للصفوف، ومؤشر للأعمدة، فمثلاً المؤشران(٤ ، ٥) يشيران إلى العنصر ٩٥، أما المصفوفة الثالثة التي بها تفاصيل الأسئلة فإن المؤشرات (٤ ،٥ ، ٣) تشير إلى درجة الطالب في السؤال الثالث في العمود الخامس، وهو الطالب رقم ٤ في قائمة الطلاب .

د لا يتم تخزين موقع كل عنصر في الذاكرة، وإنما يتم الوصول إليه بمؤشرات العنصر، وطول البعد، والموقع الأساس، فمثلاً إذا كان الموقع الأساس في المصفوفة الأولى في الذاكرة هو ٢٥٤ فإن موقع العنصر الثالث في هذه المصفوفة هو الموقع رقم ٢٥٦ أي ٢٥٤ + رقم المؤشر– ١ (لأنها بها العنصر الأول). أما المصفوفة الثانية إذا كان موقعها الأساسي ٣٥٢ فإن العنصر (٤، ٥) موقعه هو ٣٥٢ + ٤ × ٥٠ + ٣ = ٥٥٥ أي الموقع الأساسي + (المؤشر الثاني – ١) × طول البعد الأول (العمود) +

المؤشر الأول – ١ . أما العنصر الأخير أي الذي مؤشره (٥٠ ، ٩) فإن موقعه سيكون ٣٥٢ + ٨ × ٥٠ + ٤٩ = ٣٥٢ + ٤٤٩ = ٨٠١ أي أن كل العناصر مخزنة في المواقع ٣٥٢ ، ٣٥٣ ، .... ، ٨٠١ وهذه تساوي ٤٥٠موقع هي جملة عناصر المصفوفة .

مثال (٣) :

بين مواقع عناصر المصفوفة أ (٣ × ٥) الموقع الأساسي بالذاكرة = ١٣٠

المؤشرات	١،١	١،٢	١،٣	١،٤	١،٥
الموقع	١٣٠	١٣٣	١٣٦	١٣٩	١٤٢
المؤشرات	٢،١	٢،٢	٢،٣	٢،٤	٢،٥
الموقع	١٣١	١٣٤	١٣٧	١٤٠	١٤٣
المؤشرات	٣،١	٣،٢	٣،٣	٣،٤	٣،٥
الموقع	١٣٢	١٣٥	١٣٨	١٤١	١٤٤

مثلاً أ (٣ ، ٢) = ١٣٠ + ٣ (٣ – ١) + ٢ – ١ = ١٣٧

القاعدة أ (س ، ص) = الموقع الأساس + (ص – ١) × البعد الأول + س – ١

بهذه القاعدة أول عنصر أي الصف الذي في المؤشرين (١، ١) سيكون في الموقع الأساس، والعنصر الأخير الذي في المؤشرين (بعد الأول ، بعد الثاني) سيكون في الموقع الأساس + بعد الأول × (بعد الثاني – ١) + بعد الأول – ١

المصفوفات

كل لغات البرمجة تقريباً تتفق على ضرورة التعريف بالمصفوفة في بداية البرنامج، وعادة يتم تعريف اسم المصفوفة أو المتغير وعدد أبعادها وطول كل بعد، فعلى سبيل المثال تعرف المصفوفة الأولى باسمها: أسماء الطلاب، وبعدها واحد وطول البعد ٥٠، ونوع البيانات حرفية. أما المصفوفة الثانية فاسمها: درجات الطلاب، ولها بعدان، البعد الأول وطوله ٥٠، والبعد الثاني وطوله ٩، ونوع البيانات رقمية. أما المصفوفة الثالثة فلها ثلاث أبعاد، البعد الأول وطوله ٥٠، والبعد الثاني وطوله ٩، والبعد الثالث وطوله ٥ .

السـجلات

لقد سبق تعريف السجل بأنه مجموعة الحقول التي تشترك في وصف شيء واحد، وكل حقل يمثل وحدة بيانية مفردة. بهذا التعريف يمكن أن يكون السجل صفاً في مصفوفة ذات بعدين فقط الفرق بين السجل وصف المصفوفة أو بين السجلات والمصفوفات عموماً – أنه لا يشترط في السجل أن تكون كل بياناته من نوع واحد بينما يشترط ذلك في المصفوفة؛ فالسجل يمكن أن يكون بعض حقوله حرفية، وبعضها رقمية، وبعضها عددية، وبعضها منطقية. فمثلاً في المصفوفة الثانية التي بها درجات الطلاب إذا اضفنا عموداً به اسماء الطلاب وعموداً آخر به هل الطالب ناجح أو راسب– تكون المصفوفة قد تحولت إلى سجلات تحتوي على أسماء وأرقام وبيانات منطقية وبهذا التحويل لا تصلح كمصفوفة .

يتم تخزين السجلات بنفس طريقة المصفوفات، أي أن الحقول تخزن بمواقع متجاورة في الذاكرة إلا أنه ليس لحقولها مؤشرات مباشرة يتم على ضوئها حساب موقع العنصر؛ وإنما هناك معرفات أو أسماء للحقول تمثل عناوين للحقول مثلها مثل الوحدات البيانية البسيطة .

٨. الشكل العام للبرنامج في لغة باسكال :

يتألف برنامج بسكال من ثلاث أجزاء هي:

أ. رأس البرنامج :هو جزء اختياري لا يؤثر علي البرنامج ويبدأ بكلمة Program يعقبها اسم يدل علي طبيعة البرامج.

ب. جزء التعريفات أو الإعلانات : هو الجزء الذي يتم فيه الإعلان عن الجمل المستخدمة في البرنامج ويتم الإعلان عن :

- أسماء الوحدات التي من خلال الإعلان عنها نتمكن من استخدام بعض الدوال المتاحة في لغة بسكال ويتم الإعلان عن هذه الوحدات بواسطة عبارة **Uses** ويعقبها اسم الوحدة المراد الإعلان عنها فمثلاً :

```
Uses Dos;
```

فيتم من خلال هذا الإعلان التعامل مع كافة الدوال والعبارات التي تحتوي عليها الوحدة **Dos** .

- الثوابت.
- التعريفات الجديدة المستخدمة بواسطة كلمة **Type**.
- **اللافتات Label**.
- المتغيرات.
- الاجراءات المستخدمة بواسطة كلمة **Procedure** .
- الدوال المستخدمة بواسطة كلمة **Function** .

ليس من الضروري استخدام جزء الإعلانات في البرامج كله أو جزء منه ولكن حسب احتياجات البرنامج.

ج. جزء جسم البرنامج : هو الجزء الأساسي وهو ضروري لكتابة أي برنامج ويبدأ بكلمة**Begin** وينتهي بكلمة **End.** وبينهما مجموعة من العبارات التي تشكل البرنامج.

يتم في لغة باسكال أولاً تعريف البرنامج، ثم تعريف الثوابت، وتعريف المتغيرات البسيطة بأنواعها المختلفة، وتعريف المصفوفات، وتعريف السجلات . يمكن توضيح ذلك بالمثال أدناه : هذه التعريفات تبدأ أولاً بتعريف البرنامج بعد كلمة **(program)** ، ويعرف البرنامج باسمه الذي يختاره المبرمج، ثم تضع بين قوسين إن كان هناك مخرجات أو مدخلات في البرنامج أو مخرجات فقط . بعد ذلك تعرف الثوابت بعد كلمة **(const)** إذ أن الثوابت هي العناوين التي لا تقبل تغيير قيمها في الذاكرة، ولذا لابد من تعريفها للحاسوب منذ البداية . بعد الثوابت يبدأ تعريف المتغيرات، وهي إما أعداد حقيقية، أو أعداد رقمية أو منطقية أو حرفية ويتم ذلك بعد كلمة **(var)** ويحدد نوع المتغيرات بعد كتابة أسماء المتغيرات وفصلها عن بعضها بشولة وختمها بعلامة (:) .أما المصفوفات فتعرف بعد كتابة أسماء المتغيرات وعمل علامة (:) كما في حالة المتغيرات السابقة بإضافة كلمة **(array)** ، أي مصفوفة تتبعها بين قوسين بعد المصفوفة وإن كانت هناك ابعاد مركبة للمصفوفة يفصل بين البعد والآخر بعلامة شولة، وعادة يوصف البعد من ١ إلى حد البعد ثم يلي ذلك كما في حالة المتغيرات البسيطة تعرف نوع البيانات هل رقمية أم عددية أم حرفية أم منطقية .

أما تعريف السجل فأكثر تعقيداً فنبدأ أولاً بكلمة (type)، يتبعها اسم السجل ثم تساوي سجل ثم يلي ذلك تعريف الحقول، وبعد ذلك مثلما تم مع المتغيرات البسيطة، ولأن السجلات هي كما ذكرنا تمثل صفوفاً في المصفوفة كان لابد من تعريف مصفوفة تكون بياناتها من السجلات، وبهذا تكون لغة باسكال قد تحايلت على المصفوفة بجعل وحداتها البيانية من نوع بياني واحد هو السجل ولكن في الواقع يتكون السجل من أنواع شتى من البيانات، وهذه تعتبر من ميزات لغة باسكال ولغة سي .

```
Program    Example (input, output);
Uses
       Graph, Dos, Crt;
Const
       No_stud = 50;
       low_marks = 5;
       high_marks = 100;
       subj = 9;
       qu = 5;
Label
       Stop, L1 ;
Type
       Student_record  =  record
                Total : integer;
                Order : integer;
       End
Var
       root1,  root2 : real;
       Count, I  : integer;
       Found : Boolean;
       Filler : char ;
       S_names :array [1..No_stud] of char ;
       S_marks :array[1..No_stud,1..subj] of Byte ;
       S_d:array [1..No_stud,1..subj,1..qu] of Word;
       Stud_pass : array [1..no_stud] of Boolean;
       students : array [1..50] of student_record ;
       Rec :  student_record;
Procedure  proc_name;
Begin
       .
       .
End;
Function  f_name(p_var : var_type) : return_type ;
Begin
       .
       .
       func_name:=value ;
End;
Begin
       .
       .
End.
```

يمكن كتابة ذلك باللغة العربية بشكل يماثل لغة باسكال على النحو التالي :

برنامج مثال(ادخال وإخراج)
وحدات دوس ، رسم ;
ثوابت عدد الطلاب=٥٠;

أقل درجة =٥ ;
أعلي درجة ;
المواد =٩ ;
الآسئلة =٥ ;
عنوان :توقف ، ل١ ;

أنواع سجل طلاب =سجل

العدد الكلي : عدد صحيح ;
الترتيب : عدد صحيح ;
نهاية
المتغيرات الجزر١، الجزر٢: عدد حقيقي ;
العداد ، الدليل : عدد صحيح ;
موجود : بولياني ;
اسم الطالب : مصفوفة [١..عدد الطلاب] من الحروف ;
درجة الطالب : مصفوفة [١..عدد الطلاب ، ١..المواد] من الأعداد الطبيعية الصغيره ;
تفاصيل الطالب : مصفوفة [١..عدد الطلاب ، ١..المواد ،١..الاسئلة] من الأعداد الطبيعية ;
نجاح الطالب : مصفوفة [١..عدد الطلاب] من البولياني ;
طلاب : مصفوفة [١..عدد الطلاب] من سجل طلاب ;
سجل١ :من سجل طلاب ;
إجراءات اسم الاجراء ;
بداية
.
نهاية ;
دالة اسم الدالة (المتغير المرسل : نوعة) نوع القيمة الراجعة ;
بداية
اسم الدالة =: القيمة ;
نهاية ;
بداية
.
نهاية.

مثال (٤) :

اكتب شفرة توضح كيفية الإعلان عن سجل يسمي عناوين ويحتوي علي البيانات التالية :الاسم ٣٠ حرف ، الشارع ٣٠ حرف ، المدينة ٣٠ حرف
الولاية ٣٠ حرف ، الرمز رقمي

```
type address = record                    نوع العناوين = سجل
    name : array [1. .30] of char ;      الاسم : مصفوفة (١..٣٠) حرفية
    street : array [1 . . 30] of char;   الشارع : مصفوفة (١..٣٠) حرفية
    city : array [1 . . 30] of char ;    المدينة : مصفوفة (١..٣٠) حرفية
    state : array [1. .30] of char;      الولايه : مصفوفة (١..٣٠) حرفية
    code : integer;                      الرمز : رقمي
end ;                                    نهاية
```

هذا السجل يتكون من خمسة حقول كل حقل عبارة عن مصفوفة حرفية . الحقل الأول هو الاسم الذي يتكون كحد أقصى من ثلاثين حرفاً وقد أعطى كل حرف مكاناً خاصاً به في المصفوفة ليمكن من عمليات ترتيب الأسماء أبجدياً أو غير ذلك. ونفس الشي تم بالنسبة للشارع والمدينة والولاية . أما رقم أو رمز الولاية فهو رقم واحد . وحتى يتم التعامل مع هذا السجل في خزن البيانات نحتاج إلى تعريف متغير على النحو التالي :

```
Var
    students : array [1.. 50] of address;
    Line : address;
```

متغيرات

الطلاب : مصفوفة [١. .٥٠] من العناوين

عنوان : عناوين

هناك مصفوفة متغيرة أي أن بياناتها متغيرة ونوع بياناتها سجلات من نوع سجل العناوين تسمى الطلاب أو (students) وكذلك هناك متغير واحد بسيط اسمه عنوان ،ولكنه من نوع سجل العناوين فمثلاً في مصفوفة الطلاب عندما تقول "الطلاب (٢٤) . الاسم (١) " نعني الحرف الأول من اسم الطالب رقم ٢٤ في سجلات الطلاب . أما عندما تقول "عنوان . رمز" نعني قيمة الحقل رمز في السجل عنوان. أما عندما تقول "عنوان" نعني كل بيانات السجل عنوان ومثلها عندما تقول طلاب (٥) نعني كل بيانات سجل الطالب رقم ٥ في المصفوفة، وما دام السجل في المصفوفة يتم التعامل معه كاملاً كعنصر من عناصر المصفوفة فإن تخزين السجلات داخل المصفوفة يتم بنفس طريقة تخزين العناصر البسيطة في المصفوفة، مثلاً: عندما تقول الطلاب (٥٠) نعني كل سجل الطالب الأخير في قائمة الطلاب، وعندما نقول الطلاب (١) نعني كل سجل الطالب الأول، وعندما نقول الطلاب (٢) نعني كل سجل الطالب الثاني . بهذا يكون المجاور لاسم الطالب الأول هم اسم الشارع في عنوان الطالب الأول وليس اسم الطالب الثاني . أما الطالب الثاني فيجاور رمز الولاية في عنوان الطالب الأول، وهكذا . أما عدد مواقع التخزين في مصفوفة الطلاب فتساوي عدد الطلاب مضروباً في عدد العناوين أو المواقع في سجل العناوين أي تساوي ٥٠ × ١٢١ = ٦٠٥٠ موقع . يمكن وصف وضعها على النحو التالي : نفترض أن الموقع ٥٠٠ هو الموقع الأساسي بالذاكرة . هذا الموقع سيكون الحرف الأول من الاسم الأول، ثم تنتهي حروف الاسم الأول في الحرف رقم ٣٠ وهذه تأخذ عنوان مصفوفة تسمى الاسم، تليها مصفوفة الشارع التي تتكون كذلك من ٣٠ حرفاً بمعدل موقع لكل حرف ثم مصفوفة المدينة التي تتكون من ٣٠ حرفاً ثم مصفوفة الولاية التي تتكون من ٣٠ حرفاً ثم رمز الولاية الذي يأخذ موقعاً واحداً فقط فيكون مجموع المواقع لكل سجل ١٢١ موقعاً للعنصر الأول في المصفوفة (الطلاب) التي تتكون من ٥٠ سجلاً .

٩. عبارتي الاخراج والادخال في لغة باسكال :

تستخدم لغة بسكال العبارتينWriteوWriteLn لكتابة المعلومات علي الشاشة أو علي ملف محدد، والفرق الوحيد بينهما أن عبارة WriteLn تنقل المؤشر إلى سطر جديد بعد إظهار أو كتابة المعلومات (عبارات التحكم في ملحق (أ)) .

مثـــال (٥) :

```
Year:=2002;
WriteLn('Sudan');
Write('Tabat');
WriteLn(2000);
WriteLn;
Writeln('Exam',year);
```

عند تنفيذ الشفرة (جزء من البرنامج) يكون الإخراج علي النحو التالي:

```
Sudan
Tabat 2000

Exam 2002
```

معلومـة

١. يمكن أن تكون مع عبارتي الإخراج (WriteوWriteLn) الأشياء التالية:
- ثابت عددي أو سلسلي (محصور بين فاصلتين علويتين) فيكون الإخراج نفس الثابت.
- متغير فيكون الإخراج قيمة المتغير.
- تعبير جبري فيكون الإخراج قيمة التعبير الجبري.
- لاشيء فيكون الإخراج سطراً فارغاً إذا استخدمنا عبارة WriteLn .

٢. إذا كان أول متغير بعد عبارتي الإخراج (WriteوWriteLn) اسماً لملف فتتم الكتابة علي الملف وإلا سوف تتم الكتابة علي جهاز الخرج الأساسي.

تستخدم لغة بسكال العبارتين Read و ReadLn لقراءة المعلومات من لوحة المفاتيح أو من ملف حسب المتغير الأول كما ذكرنا في عبارتي الإخراج .

مثـــال (٦) :

المثال التالي يوضح الفرق بين عبارتي Readو ReadLn ، افترض أن هناك ملف يشار إلية بالمتغيرFileName يحتوي علي المعلومات التالية :

```
10   20   30    40
50
```

المطلوب ما هي قيمة المتغير **D** عند تنفيذ العبارتين التاليتين :

```
ReadLn (Filename,A,B,C);
ReadLn (Filename,D);
```

يتم إسناد القيم **10,20,30** للمتغيرات **A,B,C** علي الترتيب بواسطة الاستدعاء الأول للعبارة **ReadLn** وينتقل بعدها المؤشر الي سطر جديد بالملف وعليه فان القيمة **50** تقرأ بواسطة الاستدعاء الثاني للعبارة **ReadLn** وتسند للمتغير **D** ويتم تجاهل القيمة 40 .

وعند استبدال عبارتي **ReadLn** ب**Read**

```
Read (Filename,A,B,C);
Read (Filename,D);
```

فيتم إسناد القيم **10,20,30** للمتغيرات **A,B,C** علي الترتيب بواسطة الاستدعاء الأول للعبارة **Read** ويبقي المؤشر في نفس السطر بالملف وعليه فان القيمة **40** تقرأ بواسطة الاستدعاء الثاني للعبارة **Read** وتسند للمتغير **D** .

مثـــال (٧) :

اكتب برنامجاً بلغة بسكال يقوم بحساب مساحة الدائرة التي نصف قطرها r علمـــاً بان مساحتها تحسب وفقاً للمعادلة $area = \pi r^2$.

Const pi=3.14; **Var** **Area , r : real ;** **Begin** **Write (' أدخل قيمة نصف القطر ');** **ReadLn (r);** **Area:= pi*r*r;** **WriteLn('مساحة الدائره',area);** **End.**	**الثوابت باي=٣،١٤** **متغيرات** **مساحة ، نق: حقيقي** **البداية** **أكتب ("أدخل قيمة نصف القطر")** **أقرأ (نق)** **المساحة = باي×نق×نق** **اكتب (مساحة الدائرة = "،area)** **النهاية**

مثـــال (٨) :

اكتب برنامجاً بلغة بسكال يقوم بحساب مساحة أي مثلـــث قـــائم الزاويـــة ارتفاعه a وقاعدته b علماً بان مساحة المثلث تحسب وفقاً للمعادلة التالية

$$area = \frac{1}{2} ab .$$

Var **Area , a,b : real ;** **Begin** **Write (' أدخل قيمة طول الارتفاع ');** **ReadLn (a);** **Write (' أدخل قيمة طول القاعدة ');** **ReadLn (b);** **Area:= a*b/2;** **WriteLn('= المساحة',area);** **End.**	**المتغيرات** **المساحة ، أ ، ب :حقيقي** **البداية** **اكتب ("أدخل قيمة طول الارتفاع")** **أقرأ (أ)** **اكتب ("أدخل قيمة طول االقاعدة")** **أقرأ (ب)** **المساحة=أ×ب\٢** **أكتب ("مساحة المثلث="،المساحة)** **النهاية**

١٠. معالجات الحاسوب لأنواع البيانات في لغة باسكال :

هناك معالجات تتفق فيها كل لغات البرمجة في التعامل مع أنواع البيانات المختلفة . نذكر منها على سبيل المثال :

أ] معالجات الأعداد الحقيقية :

الضرب "*" : عدد حقيقي في عدد حقيقي أو في رقم يكون الناتج عدداً حقيقياً .

القسمة "/": عدد حقيقي على عدد حقيقي أو رقم على رقم يكون الناتج عدداً حقيقياً.

الجمع "+" : عدد حقيقي مع عدد حقيقي أو مع رقم تنتج عدداً حقيقياً .

الطرح "–" : عدد حقيقي مع عدد حقيقي أو مع رقم تنتج عدداً حقيقياً .

abs (x) :القيمة المطلقة للعدد (x) وبالطبع القيمة المطلقة لا تغير من نوع العدد .

sqr (x) : المربع لأي عدد حقيقي ينتج عدداً حقيقياً .

جا ((sin (x) وجتا ((cos (x) ومعكوس الظل ((arctan (x) واللوغريثم الطبيعي Ln (x) والقوى exp (x) الجذر التربيعي sqrt (x) كلها دوال تنتج عدداً حقيقياً سواء أكانت قيمة x رقماً أم عدداً .

ب]معالجات الأعداد الرقمية :

الضرب "*" : ضرب الأرقام (عدد صحيح) ينتج أرقاماً .

القسمة "/" : قسمة الأرقام لا تنتج أرقاماً بل تنتج أعداد حقيقية .

الجمع "+" : الجمع للأرقام ينتج أرقاماً .

الطرح "–" : الطرح للأرقام ينتج أرقاماً .

y DIV x : هذه قسمة تنتج رقماً بكشط الكسر الناتج بعد قسمة y على x

y mod x : هذه تنتج رقما هو باقي قسمة y على x .

Sqr (x) : هذه دالة المربع وبالطبع تنتج رقماً إذا كان x رقماً .

Trunc (x): هذه تنتج من العدد الحقيقي x رقماً بكشط الكسر .

Round (x): هذه تنتج من العدد الحقيقي x رقماً بتقريب الكسر .

ج] معالجات الحروف :

نعني بالحروف في لغة باسكال الحروف الهجائية, والأرقام ،وحرف الفراغ. هناك دالتان في لغة باسكال وفي غالب لغات البرمجة تتعامل مع البيانات الحرفية وهي الدالة **ord(c)** والدالة **chr(i)** . وتعنى الدالة الأولى **ord(c)** ترتيب رمز الحرف **(c)** من جدول ترميز الحروف ، بينما الدالة الثانية **chr(i)** تعنى الحرف الذي رمزه يساوي **(i)** . وبما أن الدالة **ord(c)** ناتجها رقم فإنه بالإمكان إجراء أو تطبيق كل العمليات التي تتم على الأرقام عليها ، مثل عمليات أكبر من، وأصغر من، ويساوي وكل تركيباتها لأن في ذلك أهميته في التساؤل والاستفسار عن ترتيب الأسماء مثلاً .

الدالة **ord(c)** والدالة **chr(i)** متعاكستان أي أن:

. ord(chr(i)) = i و chr(ord(c))= c

د] معالجات البيانات المنطقية :

AND (تحقق كل الشروط) ينتج عنها قيمة منطقية .

OR (تحقق أي شرط) ينتج منها قيمة منطقية .

Not (نفي الشرط) ينتج منها قيمة منطقية .

أقل من < ينتج منها قيمة منطقية .

أقل من أو يساوي <= ينتج منه قيمة منطقية .

يساوي = تنتج منه قيمة منطقية .

لا يساوي <> ينتج منه قيمة منطقية .

Odd (x) ينتج منه قيمة منطقية وهي تحقق فردية x .

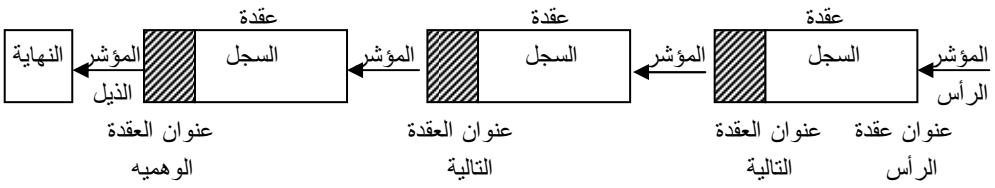
Eol (f) ينتج منه قيمة منطقية وهي تحقق انتهاء السطر .

Eof (f) ينتج منه قيمة منطقية وهي انتهاء الملف .

١١. البنائيات المتجردة :

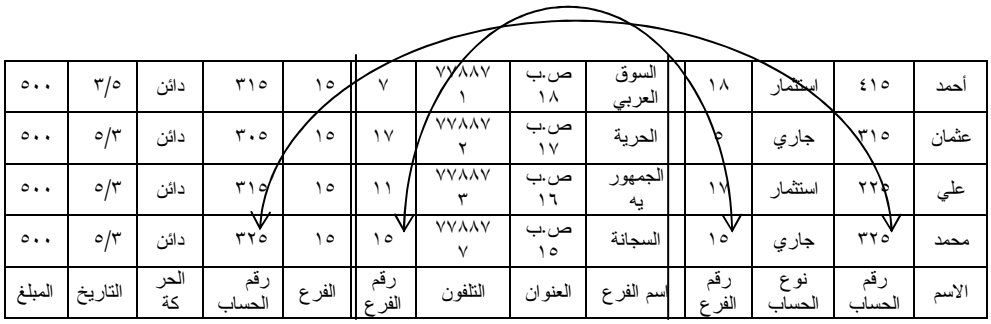
لقد رأينا في الفقرات السابقة أن لغات البرمجة تحتوي على بعض البنائيات البيانية البسيطة، أو المتغيرات البسيطة، والبنائيات المركبة مثل المصفوفات ومصفوفات السجلات ولكن هناك بنائيات أكثر تعقيداً وتستطيع أن تتعامل مع بعض التطبيقات بكفاءة عالية ولكنها غير موجودة في لغات البرمجة؛ لأنها تحتاج إلى برمجيات خاصة تمكن من التعامل معها . هذه البنائيات تعرف بالبنائيات المتجردة. إذن البنائية المتجردة هي عبارة عن بنائية بيانية زائداً البريمجات التي تمكن من التعامل معها . ولما كانت العمليات التي تتم على البنائية البيانية هي في الغالب إضافة وحدة بيانية أو حذفها؛ لذا تصمم البنائية المتجردة عادة لتكون مناسبة للطريقة التي يتم بها الإضافة والحذف مثلاً البنائية التي تعرف بالمكدسات **(stacks)** يتم فيها الحذف والإضافة من اتجاه واحد مثل طريقة التخزين عموماً حيث يتم سحب الأشياء التي وصلت أخيراً قبل السابقة، وهنا يعرف بنظام ليفو **(lifo)** لتكدس الأشياء بعضها فوق بعض وليس بالإمكان سحب السفلي قبل العليا . إذن بنائية المكدس تناسب تطبيقات المخازن عموماً . أما المثال الثاني للبنائيات المتجردة فهو البنائية التي تعرف بالصفوف **(queues)** ، والصف يتم فيه الإضافة والحذف من اتجاهين متعاكسين مثل الصف تماماً، حيث أن أول من وصل هو أول من يخرج وهنا يعرف بنظام فيفو **(fifo)** وبنائية الصفوف تناسب كل تطبيقات الصفوف التي عادة ما يأخذ فيها الخدمة السابق قبل اللاحق . هناك بنائية متجردة من أكثر البنائيات المتجردة مرونة تعرف بالقوائم المتصلة، وهي عبارة عن سجلات كل سجل له تكوينه الخاص . ليس بالضرورة أن تأخذ كل السجلات شكلاً واحداً . الشكل العام للسجل أن يكون به حقل يخزن فيه عنوان وموقع السجل التالي له في الذاكرة، إضافة إلى حقل واحد أو أكثر يحوي الوحدات البيانية في السجل . يعرف السجل في القوائم المتصلة بالعقدة **(node)** ويعرف حقل موقع السجل التالي بالمؤشر . هناك عقدتان أساسيتان في القوائم المتصلة هي العقدة الأولى التي ليس هناك عقدة تحفظ عنوانها أو تشير إليها، وتعرف برأس

القائمة ،والعقدة الأخيرة التي ليس بها عنوان عقدة أو لا تشير لعقدة تالية لها تعرف بذيل القائمة، ولكن لأن بالضرورة هناك عنواناً في أي عقدة فإن ذيل القائمة تشير إلى عقدة وهمية تعني نهاية القائمة. هذه البنائية تناسب الملفات متغيرة السجلات .



القوائم المتصلة

إضافة إلى هذه الأمثلة الثلاث يمكن اعتبار قاعدة البيانات التي يتم تصميمها بواسطة برمجيات قواعد البيانات مثل برنامج إس كيو إل **(sql)** أو أوراكل أو فوكس برو بنائية متجردة، وهي تناسب المجموعات البيانية الضخمة والمتداخلة مثل بيانات عملاء البنك الأساسية، وحركة حساباتهم في الحسابات المختلفة، كالحساب الجاري والاستثمار وغيره، وتعاملهم في الأفرع المختلفة، ومقاضاتهم مع البنوك الأخرى داخلياً وخارجياً . ومثال آخر بيانات حركة الطيران في الخطوط المختلفة وحجوزات العملاء، وبيانات المسافرين، أمتعتهم، وحركة البضائع. ومثال آخر بيانات الشرطة التي تشمل البطاقة الشخصية، والجنسية، والجوازات، ودخول وخروج المواطنين والأجانب، وملفات الجرائم والمشبوهين وغيرها . إن كل الأنظمة التي تتعامل مع بيانات ضخمة ومختلفة ومتداخلة تناسبها تطبيقات البنائية المتجردة التي تعرف بقواعد البيانات العلائقية، وهي ربط بين منطقي وعملي بين البنائيات المتجردة وقواعد البيانات. فمثلاً في المثال السابق بيانات العملاء تمثل قاعدة بيانات، وبيانات المسافرين تمثل قاعدة بيانات، وبيانات البضائع تمثل قاعدة بيانات وحركة الطيران تمثل قاعدة بيانات، وهكذا. وكلها عند ربطها مع بعضها تمثل بنائية متجردة معقدة تعرف باسم قواعد البيانات العلائقية . ومثال لذلك العلائقية التالية :



الاسم	رقم الحساب	نوع الحساب	رقم الفرع	اسم الفرع	العنوان	التلفون	رقم الفرع	الفرع	رقم الحساب	الحركة	التاريخ	المبلغ
أحمد	٤١٥	استثمار	١٨	السوق العربي	ص.ب ١٨	٧٧٨٨٧١	٧	١٥	٣١٥	دائن	٣/٥	٥٠٠
عثمان	٣١٥	جاري	٥	الحرية	ص.ب ١٧	٧٧٨٨٧٢	١٧	١٥	٣٠٥	دائن	٥/٣	٥٠٠
علي	٢٢٥	استثمار	١١	الجمهورية	ص.ب ١٦	٧٧٨٨٧٣	١١	١٥	٣١٥	دائن	٥/٣	٥٠٠
محمد	٣٢٥	جاري	١٥	السجانة	ص.ب ١٥	٧٧٨٨٧٧	١٥	١٥	٣٢٥	دائن	٥/٣	٥٠٠

هذه العلائقية تفيد أنه يوم ٥/٣ حدثت حركة في الحساب رقم ٣٢٥ بمبلغ ٥٠٠ دينار وهذا الحساب يخص محمد بفرع السجانة وهو حساب جاري وعنوان فرع السجانة هو ص.ب ١٥ وتلفون ٧٧٨٨٧٧ .

١٢. خوارزميات الإضافة والحذف :

يوجد نوعان من الخوارزميات هما :

(أ) خوارزمية الإضافة

(ب) خوارزمية الحذف .

أ] خوارزمية الإضافة :

١. هل الموقع الأعلى أصبح يساوي أو أكبر من حجم المكدس؟ إن كان "نعم" لا يمكن الإضافة وتنتهي الخوارزمية (تخرج الرسالة لا يمكن الإضافة لغمر المكدس) وإن كان "لا" تستمر الخوارزمية في إضافة العنصر .
٢. الموقع الأعلى=الموقع الأعلى+١ (حرك الموقع الأعلي خطوه الي أعلي)
٣. المصفوفة (الأعلى) = العنصر المضاف .

أي يضاف العنصر الي المصفوفة في الموقع الذي رقمه يساوي الرقم الأعلى السابق زائداً واحد .

فيما يلي برنامج خوارزمية الإضافة :

برنامج إضافة عنصر للمكدس
ثابت
الأكبر = ١٠٠
متغيرات
الموقع الأعلى :الرقمية
العنصر المضاف : حرفي
المكدس : مصفوفة[١ .. الأكبر]
حرفي
أبدأ
أقرأ (الموقع العلي)
أقرأ (العنصر المضاف)
إذا كان موقع الاعلي=الاكبر
اكتب رسالة(المكدس مغمور)
والا
البداية
الموقع الأعلى = الموقع الأعلى + ١
المكدس (الموقع الأعلى) = العنصر المضاف
انتهت إلا
النهاية .
	Program add_stack;
	Const
	max = 100;
	Var
	Top : integer;
	Newlelement: char;
	stack:array[1..max] of char;
	Begin
	Readln(top);
	Readln(newelement);
	*** If top = max then**
	writeln ('over flow')
	Else
	Begin
	Top : = top + l;
	Stack(top)=newelement;
	End;
	End.

ب] خوارزمية الحذف :

١. هل الموقع الأعلى = صفر إن كان "نعم" لا يمكن الحذف وتنتهي الخوارزمية وتخرج الرسالة المكدس فارغ وإن كان "لا" ، تستمر الخوارزمية في عملية الحذف .
٢. العنصر المحذوف = المصفوفة (الموقع الأعلى) أي يحذف العنصر الذي رقمه في المصفوفة = الموقع الأعلى .
٣. الموقع الأعلى = الموقع الأعلى – ١ .

---

*عبارات التحكم في لغة بسكال موضحة في الملاحق

فيما يلي برنامج خوارزمية الحذف :

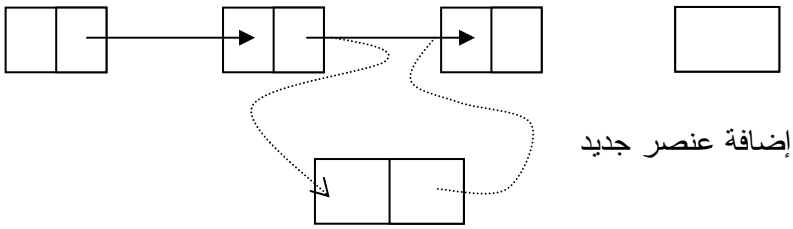
برنامج حذف عنصر المكدس
ثابت
الأكبر = ١٠٠
متغيرات
الموقع الأعلي : رقمي
العنصر المحذوف : حرفي
المكدس : مصفوفة (١.. الأكبر)
حرفي
أبدأ
ادخل قيمة الموقع الأعلى الحالية
إذا كانت قيمة الأعلى = صفر
اكتب (المكدس فارغ)
وإلا
أبدأ
العنصر المحذوف = المكدس (الموقع الأعلى)
الموقع الأعلى=الموقع الأعلى – ١
انتهت إلى النهاية.
	Program delete stack;
	Const
	max = 100;
	Var
	top : integer;
	Deleteelement : char;
	stack:array[1..max] of char;
	begin
	readLn [top] ;
	If top = 0 then
	writeln ('stack empty')
	else
	begin
	Deleteelement:=stack(top);
	Top : = top – 1;
	end ;
	end .

١٣. مقارنة القوائم المتصلة والمصفوفات :

أولاً : تميزت القوائم المتصلة على المصفوفات بديناميكية مساحة التخزين حيث أن المساحة تتقلص أو تزيد كلما حذف عنصر أو أضيف عنصر، أما في المصفوفات فمنذ البداية لابد من حجز كل المساحة؛ إذ أنه لا مجال لتغيير مساحة المصفوفة لأنها منذ البداية يتم تعريفها وفق مساحة ثابتة ومواقع ثابتة لكل عناصر المصفوفة في الذاكرة .

ثانياً: تميزت القوائم المتصلة بسهولة الحذف والإضافة حيث يتم كل ذلك فقط بتغيير مؤشر الرأس هذا في حالة المكدس، أما في حالة أخرى فإنه بالإمكان

إضافة أي عنصر إلى موقع من مواقع القائمة فقط بجعل مؤشر العنصر الجديد يؤشر إلى العنصر السابق له، ومؤشر العنصر اللاحق له يشير إليه كما في الرسم



إضافة عنصر جديد

إذا كانت السجلات مرتبة فإنه يمكن إضافة إي عنصر ووضعه في مكانه الصحيح حسب الترتيب. أما في المصفوفات فلا يتم ذلك بهذه السهولة حيث يتم تحريك كل العناصر الأصغر خطوة نحو الخلف، أو كل العناصر الأكبر خطوة نحو الأمام .

ثالثاً : تميزت المصفوفات على القوائم المتصلة عند البحث عن معلومة أو الوصول إلى معلومة معينة، فيتم ذلك بطرق شتى ، وليس هناك إلا طريقة واحدة فقط بالنسبة للقوائم المتصلة، وهي طريقة البحث المتتالي أي أن البحث عن المعلومة يتم مبتدئاً من رأس القائمة حتى نجد المعلومة المطلوبة .

١٤. تمرين

١/ عرف بنائيات البيانات.

٢/ عرف الكلمة والعنوان فى ذاكرة الحاسوب.

٣/ عرف عناوين الذاكرة والثابتة وعناوين الذاكرة المتغيرة.

٤/ تتفق كل لغات البرمجة على أربع أنواع من البيانات سمها وصفها.

٥/ ما دور جدول الترميز فى ذاكرة الحاسوب.؟

٦/ ما العنوان البسيط والعنوان المركب فى ذاكرة الحاسوب ؟

٧/ كيف يعرف السالب والموجب فى لغات البرمجة؟

٨/ كيف يمثل الرقم الموجب ٧؟

٩/ كيف يمثل الرقم السالب ٧ بخوارزمية الاكمال الثنائى والاكمال الأحادي؟

١٠/ ما تمثيل سالب صفر العلامة والقيمة. الإكمال الثنائى والاكمال الاحادي مقارنة بتمثيل موجب صفر؟

١١/ أجمع سالب ٥ مع موجب ٧ بالطرق الثلاث وبين ملاحظاتك .

١٢/ مثل العدد الحقيقي ٢٢٧,٩٨ في حاسوب طول كلمته ٢٤ ثنائية وحدد المانتسا والأس.

١٣/ أجمع ٠,٠٢٧ مع ٧٨٩٨٥٣٢ فى حاسوب طول كلمته١٨ ثنائية.

١٤/ ما أهم نظم تمثيل الحروف ومالفرق بينهما؟

١٥/ كيف يتم تمثيل البيانات المنطقيه فى الحاسوب؟

١٦/ كيف يتم تحويل الحرف الى رقم والرقم الى عدد والعدد الى رقم؟

١٧/ كيف يتم تعريف نوع البيانات فى لغات سى وباسكال؟

١٨/ عرف المصفوفة.

١٩/ أعط مثالاً لمصفوفة ذات بعد واحد عن أسماء السلع وذات بعدين عن رقم السلعة وثمن السلعة. سم المصفوفة الأولى السلع والمصفوفه الثانية تجارة السلع ثم أقرأ تجارة السلع (٣،٣) وتجارة السلع (٤،٢) .

٢٠/ ما موقع الأساس في المصفوفة ؟

٢١/ ما معادلة موقع العنصر فى مصفوفه ذات بعد واحد وذات بعدين؟ ثم حدد فى المثال السابق موقع السلع (٢) وموقع تجارة السلع (٤،٢).

٢٢/ كيف تعرف المصفوفة في لغات البرمجة ؟

٢٣/ ما الفرق بين السجل والمصفوفه ذات البعدين؟ ومن ثم عدل مثال تجارة السلع ليصبح سجلا بدلا من مصفوفه ذات بعدين.

٢٤/ كيف يتم معرفة العناصر فى السجلات؟

٢٥/ أعط مثالاً لتعريف المصفوفة والسجل في لغة باسكال ومن ثم مصفوفة السجلات.

٢٦/ صف معالجات الاعداد الحقيقيه فى لغة باسكال.

٢٧/ صف معالجات الأعداد الرقمية فى لغة باسكال.

٢٨/ صف معالجات الحروف فى لغة باسكال.

٢٩/ صف معالجات البيانات المنطقية فى لغة باسكال.

٣٠/ عرف البنائيات المتجردة.

٣١/ صف بنائية المكدسات . أعط أمثله لتطبيقاتها.

٣٢/ صف بنائية القوائم المتصله واعط أمثله لتطبيقاتها.

٣٣/ صف بنائية قواعد البيانات وأعط أمثله لتطبيقاتها وبرمجياتها.

٣٤/ صف خوارزمية الحذف والإضافة فى المكدس باستخدام المصفوفة.

٣٥/ صف خوارزمية الحذف والإضافة فى المكدس باستخدام القوائم المتصلة.

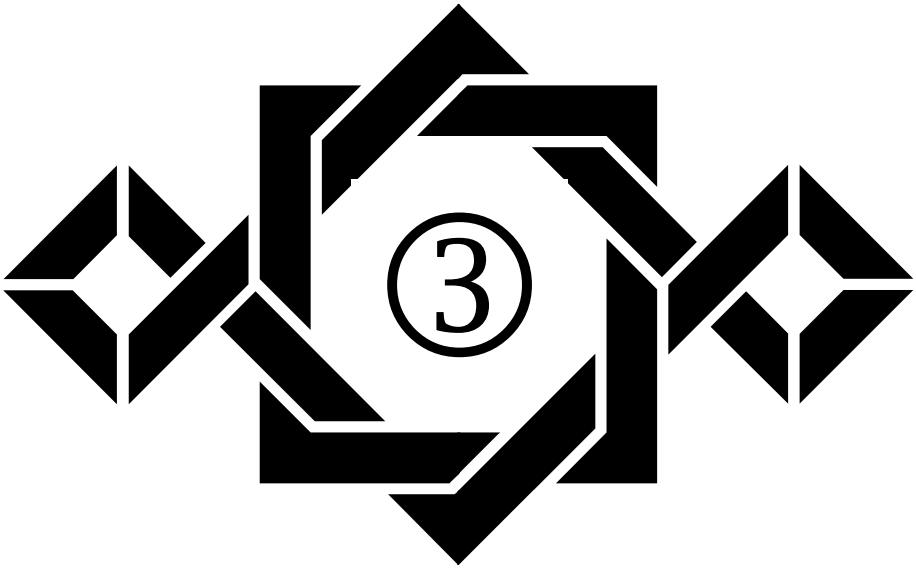
٣٦/ قارن بين بنائية المصفوفه وبنائية القوائم المتصلة.

٣٧/ أكتب برنامجا يشبه باسكال لحذف وإضافة عنصر مكدس مستخدما المصفوفه والقوائم المتصلة.

٣٨/ أكتب برنامجاً يشبه بسكال يقوم بحساب مساحة أي مثلث أضلاعه a,b,c علماً بان مساحة المثلث تحسب وفقاً للمعادلة التالية:

$$area = \sqrt{s(s-a)(s-b)(s-c)}.$$

حيث $s = \frac{a+b+c}{2}$



الخوارزميّات البيانيّة

الخوارزميات البيانية

١. خوارزميات البحث عن المعلومة :

بالطبع من الضروري أن نعرف أين أماكن تخزين المعلومات حتى نستطيع الوصول إليها بغرض التعامل معها سواء أكان ذلك لمعرفتها أم تحديثها أم حذفها أم غير ذلك. لقد رأينا أن أكثر البنائيات مرونة في استيعاب أنواع شتى من البيانات وبأحجام كبيرة هي البنائيات التي تمت عبر مصفوفات السجلات . لذا عندما نتحدث عن خوارزميات البحث عن المعلومة فإننا نتحدث عن خوارزميات البحث عن سجل في مصفوفة السجلات ثم البحث عن المعلومة في أحد حقول ذلك السجل ويتم ذلك عادة بالبحث عبر حقل في السجل للوصول لذلك السجل، ثم بعد ذلك يتم التعامل مع أي حقل آخر من حقول السجل أو مع كل حقول ذلك السجل . على سبيل المثال يمكن أن يكون :

- حقل رقم الطالب هو الحقل الذي يتم به البحث عن سجل الطالب وبعد الوصول إلى سجل الطالب يمكن النظر إلى حقل ترتيب الطالب .

- حقل رقم الموظف هو الحقل الذي يتم به البحث عن سجل الموظف ثم بعد ذلك يتم النظر إلى راتب الموظف إن كنا نريد معرفة راتبه أو تعديله أو النظر إلى تاريخ تعيينه .

- حقل اسم الشخص هو الحقل الذي يتم به البحث في سجلات التلفونات لمعرفة تلفون شخص معين وهذه هي الطريقة الطبيعية لمعرفة تلفون معين في دليل التلفونات .

في هذه الأمثلة يسمى حقل رقم الطالب في سجلات الطلاب، وحقل رقم الموظف في سجل الموظفين، وحقل اسم الشخص في دليل التلفونات بالمفتاح .

هناك خوارزميات شتى تم تصميمها للبحث عن المعلومة من أشهرها خوارزمية البحث المتتالي أو البحث الخطي وخوارزمية البحث الثنائي وغيرها . لغرض هذا المنهج سوف نكتفي بوصف خوارزمية البحث المتتالي وخوارزمية البحث الثنائي .

٣. خوارزمية البحث المتتالي :

هذه الخوارزمية تنظر ببساطة إلى صف المفاتيح على التوالي من أول مفتاح إلى آخر مفتاح وفي كل مرة تقارن بين المفتاح الذي في الصف وبين المفتاح المطلوب حتى نجد المفتاح المطلوب أو ينتهي الصف.

مثال (١) :

إذا كان صف المفاتيح هو :

مؤشر المصفوفة	١	٢	٣	٤	٥	٦	٧	٨
المفتاح	١١٠	١١٢	١٥٦	٢١٠	٢٥٧	٢٩٦	٣٥٠	٨٩٢

والمفتاح المطلوب هو ٣٥٠ فإن الخوارزمية تبدأ بمقارنة ٣٥٠ مع :

- المفتاح الأول ١١٠ فلا نجد أنه يساوي المفتاح المطلوب ٣٥٠ .
- المفتاح الثاني ١١٢ فلا نجد أنه يساوي المفتاح المطلوب ٣٥٠ .
- المفتاح الثالث ١٥٦ فلا نجد أنه يساوي المفتاح المطلوب ٣٥٠.

- المفتاح الرابع ٢١٠ فلا نجد أنه يساوي المفتاح المطلوب ٣٥٠ .
- المفتاح الخامس ٢٥٧ فلا نجد أنه يساوي المفتاح المطلوب ٣٥٠ .
- المفتاح السادس ٢٩٦ فلا نجد أنه يساوي المفتاح المطلوب ٣٥٠.
- المفتاح السابع حيث نجد أنه يساوي المفتاح المطلوب ٣٥٠ .

إننا عندما نتحدث عن كفاءة أو سرعة هذه الخوارزمية، فإننا نجد أن متوسط عدد المقارنات =(ن+١)\٢ إذا كانت ن هي عدد المفاتيح باعتبار إذا كان المفتاح الأول هو المفتاح المطلوب فإن الخوارزمية تكون قد اكتفت بمقارنة واحدة فقط وهي أحسن حالة ممكنة وأما أسوأ حالة هي أن يكون المفتاح الأخير هو المفتاح المطلوب وعليه يكون في هذه الحاله عدد المقارنات يساوي "ن". هذا النوع من الخوارزميات يوصف عدد مقارناته بأنها تزداد طردياً مع عدد المفاتيح .

الخوارزمية :

١. ادخل عدد المفاتيح ن ومصفوفة المفاتيح .
٢. ادخل المفتاح المطلوب .
٣. اجعل عداد المفاتيح في البدء = ٠.
٤. عداد المفاتيح = عداد المفاتيح + ١.
٥. اقرأ مفتاح المصفوفة الذي يشير إلية المؤشر.
٦. هل تطابق المفتاح مع مفتاح المصفوفة ؟ نعم .اذهب الي ٨ .
٧. هل عداد المفاتيح أكبر من ن؟ " لا" اذهب إلى ٤. "نعم" اذهب الي١٠.

٨. أكتب "وجد المفتاح في" عداد المؤشر.

٩. اذهب الي ١١.

١٠. اكتب "لم يوجد المفتاح".

١١. النهاية.

```
Program searshline ;
const
      N: = 50;
Var
    I, key: integer ;
    found = Boolean ;
    Keys = array [1..n] of  integer ;
Begin
   for i:=1 to n do
      readln(keys(I)) ;
   I : = 0 ;
   found : = false;
   readln (key);
   repeat
     I : = I+1;
     If keys (I) = key then
           found : = true;
   Until  I > n or found =true;
   If  I>n then
      write ('key not found')
   Else
      write ('key found in' ,I);
End.
```

برنامج البحث الخطي

ثابت

ن = ٥٠ عدد المفاتيح

المتغيرات

المؤشر ، المفتاح : عدد صحيح

هل وجد : منطقي

المفاتيح مصفوفة من ١ إلى ن : رقمية

أبدأ

من المؤشر=١ إلى ن

أقرأ المفاتيح (المؤشر)

المؤشر= ٠

المفتاح لم يوجد

أقرأ المفتاح

كرر

المؤشر = المؤشر + ١ ،

إذا تطابق المفتاح مع المفتاح الذي يشير إليه المؤشر

اجعل المتغير المنطقي هل وجد؟ نعم

اوقف التكرار إذا كانت الإجابة نعم أو زاد عدد المؤشر على ن

إذا كان عدد المؤشر أكثر من ن

اكتب لم يجد المفتاح

وإلا

اكتب وجد المفتاح في المؤشر

النهاية

٣. خوارزمية البحث الثنائي :

هذه الخوارزمية تفترض أن المفاتيح مرتبة ترتيباً تصاعدياً ثم بعدها تقارن المفتاح المطلوب مع المفتاح الذي في وسط القائمة إي المفتاح الذي يؤشر إليه المؤشر رقم و=(ن+١)\٢ باعتبار "ن" هي عدد المفاتيح (إذا كان هناك كسر في القسمة يتم إسقاطه) فإذا كان المفتاح المطلوب أكبر من مفتاح الوسط "و" فإننا نهمل البحث في النصف الذي به المفاتيح الأصغر من المفتاح المطلوب، أي بلغه أخرى نهمل كل المفاتيح من المؤشر رقم ١ إلى المؤشر "و" . نجعل بعد ذلك المؤشر رقم "و" هو مثل المؤشر رقم ١ في المعالجة السابقة ونحسب المؤشر الوسط في النصف من المؤشر رقم "و" إلى المؤشر رقم "ن" والذي يساوي (و+ن)\٢ هي قيمة مؤشر الوسط الجديد "وج" ثم تقارن المفتاح الذي يؤشر إليه مؤشر الوسط الجديد مع المفتاح المطلوب، فإن كان المفتاح المطلوب هذه المرة أصغر من مفتاح الوسط الجديد فإننا سنهمل الربع الأكبر من مفتاح الوسط الجديد وننظر فقط للربع من مفتاح الوسط الجديد "وج" إلى مفتاح الوسط "و" ثم نكرر العملية ونحسب مؤشر الوسط الأجد "وأ" والذي يساوي (و+وج)\٢ ثم تقارن المفتاح المطلوب مع مفتاح الوسط الأجد، فإن كان أكبر منه نظرنا إلى الثمن من مفتاح الوسط الأجد إلى مفتاح الوسط الجديد ويستمر هكذا نقسم النصف إلى نصفين إلى أن نبقى على مفتاح واحد هو المفتاح المطلوب .

إذا لم نجد مفتاحاً مساوياً للمفتاح المطلوب يكون في هذه الحالة المفتاح المطلوب غير موجود .

المثال التالي يمكن أن يوضح الخطوات على التوالي :

مثال (٢) :

إذا كان صف المفاتيح هو :

المفتاح	١١٠	١١٢	١٥٦	٢١٠	٢٥٧	٢٩٦	٣٥٠	٨٩٢

والمفتاح المطلوب هو ٣٥٠ فإن الخوارزمية تبدأ ب:

القائمة الأولي ١١٠ ١١٢ ١٥٦ ٢١٠ ٢٥٧ ٢٩٦ ٣٥٠ ٨٩٢

الخطوة الأولى مفتاح الوسط = ٢١٠ المفتاح المطلوب ٣٥٠ > ٢١٠

القائمة الثانية ٢١٠ ٢٥٧ ٢٩٦ ٣٥٠ ٨٩٢

الخطوة الثانية مفتاح الوسط ٢٩٦ المفتاح المطلوب ٣٥٠> ٢٩٦

القائمة الثالثة ٢٩٦ ٣٥٠ ٨٩٢

الخطوة الثالثة مفتاح الوسط ٣٥٠ المفتاح المطلوب ٣٥٠ = ٣٥٠

المفتاح المطلوب هو الذي يؤشر إليه المؤشر رقم ٧

الخوارزمية :

١- ادخل المفتاح المطلوب وقائمة المفاتيح مرتبة تصاعدياً حيث أن المؤشر رقم ١ يؤشر لأصغر مفتاح والمؤشر الأخير "ن" يؤشر لأكبر مفتاح (ن عدد المفاتيح).

٢- هل المؤشر الأول = المؤشر الأخير ؟ نعم اذهب ٦.

٣- احسب مؤشر الوسط و = (المؤشر الأول + المؤشر الأخير)\٢.

٤- هل مفتاح المؤشر الوسط = المفتاح المطلوب ؟ نعم اذهب إلى ٨ .

٥- إذا كان مفتاح المؤشر الوسط > المفتاح المطلوب اجعل المؤشر الأخير = المؤشر الوسط وإلا اجعل المؤشر الأول = المؤشر الوسط .

٦- أكتب لا يوجد المفتاح .

٧- اذهب إلى ٩.

٨- أكتب المؤشر الوسط .

٩- النهاية.

نلاحظ في هذه الخوارزمية أن عدد المقارنات يساوي $لو_{٢}ن$ إذا افترضنا مثلاً أن ن = ١٦ فإن الخطوة الأولى سيكون عدد المفاتيح ٨ ثم الخطوة الثانية يكون عدد المفاتيح ٤ ثم الخطوة الثالثة يكون عدد المفاتيح ٢ ثم الخطوة الرابعة يكون عدد المفاتيح ١ وهو المفتاح المطلوب . إذن تمت ٤ مقارنات حتى وصلنا إلى المفتاح (٤ هي في الواقع $لو_{٢}١٦$).

ملحوظة :

إننا في مثالي وبرنامجي خوارزمية البحث المتتالي أو الخطي والبحث الثنائي أو الزوجي- جعلنا مصفوفة المفاتيح رقمية، وهذا فقط على سبيل المثال؛ لأن المفاتيح يمكن أن تكون حرفية مثل الأسماء، أو أن تكون أي نوع آخر. ولكن في غالب الاستخدامات تكون المفاتيح رقمية لرفع كفاءة البحث؛ لأن مقارنة الأرقام أسرع من مقارنة الأنواع الأخرى؛ لذا دائماً يستحسن أن يتم البحث برقم الموظف، ورقم الطالب بدلاً من اسم الموظف، واسم الطالب بحيث يكون هناك رقم مفرد لأي طالب، ولآي موظف على سبيل المثال . نجد عموماً أن أغلب التصميمات تعطي مفتاحاً رقمياً مفرداً للمتعاملين مثل رقم الحساب ورقم بطاقة الخدمة الوطنية .. كما لابد من الإشارة إلى أن هذا المفتاح يشير إلى معان هامة مثل رقم فرع البنك، ورقم البنك، ثم رقم العميل، وكذلك رقم البطاقة الشخصية يشير إلى رقم الولاية ورقم المحافظة ورقم المحلية ثم رقم المواطن مثلاً وهكذا .

```
Program binary_search;
Const
   n: = 50;
Var
    I,Key, first, last, middle: integer;
    Found : Boolean;
    Keys : array [1..n] of integer;
Begin
  for I:=1 to n do
     readln(keys(I)) ;
     readln(key);
     Found : = false;
     first : = 1;
     last: = n;
     Repeat
         Middle:= (first + last) Div 2;
         If  key[middle] > key then
         Begin
            last: = middle ;
            If key [middle] < key then
                First: = middle
            Else
                found := true;
         End
     Until found or last <= first
     If found then
        Writeln('key found', middle)
     else
        writeln ('key not found');
End.
```

برنامج البحث الثنائي

الثوابت

ن = ٥٠ عدد المفاتيح

المتغيرات

المفتاح ، الأول ، الأخير ، المؤشر الوسط : عدد صحيح

هل وجد : منطقية

مفاتيح مصفوفة من ١ إلى ن رقمية

أبدأ

من ١ إلى ن

اقرأ عناصر المصفوفة

أقرأ المفتاح

هل وجد= لا

المؤشر الأول = ١

المؤشر الأخير = ن

كرر

المؤشر الوسط = إسقاط((المؤشر الأول + الأخير)\٢)

إذا كان مفتاح المؤشر الأوسط أكبر من المفتاح

اجعل المؤشر الأخير = المؤشر الوسط

إذا كان مفتاح المؤشر الأوسط أصغر من المفتاح

أجعل المؤشر الأول = المؤشر الوسط

وإلا

اجعل هل وجد = نعم

حتي يتحقق وجد المفتاح أو المفتاح الأخير يساوي أ و أقل من المفتاح الأول

إذا هل وجد = نعم اطبع "وجد المفتاح في المؤشر الأوسط" وإلا اطبع "لم يوجد المفتاح"

النهاية

٤. تصنيف المعلومات :

من الاستخدامات المتكررة في معالجة البيانات تصنيف المعلومات، ومعنى تصنيف المعلومات وضع السجلات حسب ترتيب معين ويتم ذلك بالطبع عن طريق ترتيب مفاتيح السجلات التي ترتب ترتيباً تصاعدياً أو تنازلياً .

فإذا كان المفتاح مفتاحاً رقمياً مثل رقم الطالب أو رقم الموظف أو رقم البطاقة الشخصية فإن ترتيب المفاتيح يتم من أصغر رقم أي أصغر قيمة مفتاح تصاعدياً إلى أكبر رقم أي أكبر قيمة مفتاح وبهذا تكون سجلات الطلاب أو الموظفين أو المواطنين تم ترتيبها تصاعدياً، والعكس إذا بدأنا الترتيب بأكبر رقم ثم نزلنا لأصغر يكون الترتيب تنازلياً، وبالمثل إذا كان المفتاح بالأسماء فإن الترتيب يكون أبجدياً مبتدئاً بالحرف الأول للاسم بالألف ومنتهياً بالياء وعندما يتساوى مفتاحان في الحرف الأول ينظر إلى الحرف الثاني وإذا تساوى في الحرف الثاني ينظر إلى الحرف الثالث وهكذا . ومن الضروري ألا يتطابق اسمان في كل الحروف وإلا يكون حقل الاسم غير صالحاً ليكون مفتاحاً، لأن هنالك شرطاً أساسياً لمفتاح السجل وهو أن يكون مفرداً في التعبير عن السجل (لهذه المشكلة لا يفضل استخدام الأسماء في المفاتيح).

إن أهم استخدامات التصنيف كما رأينا في الخوارزمية هو تسهيل أو تسريع عملية البحث عن السجل هذا إضافة إلى استخدامات أخرى في ترتيب المستويات للأفراد مثل نتائج الامتحانات والمنافسات بين الطلاب والموظفين عامة المتنافسين ومثل أولويات الحجوزات والطلبات والخدمات عموماً والتي تعطي أولوية للأول فالتالي وهكذا . إذن ليس كل الترتيب يتم فقط لحقل المفاتيح وإنما يمكن أن يتم على أي حقل حسب الحاجة الاستفسارية أو التحليلية .

هناك خوارزميات عدة في تصنيف المعلومات بعضها معقدة جداً وعالية الكفاءة ونعني بالكفاءة أن يتم التصنيف في أقل وقت ممكن وفي أقل سعة تخزينية ممكنة ولكن لغرض هذا المنهج سوف نركز على نوعين من الخوارزميات يتميز بالبساطة وكثرة الاستخدام .

النوع الأول يعرف بخوارزميات التبديل وأشهرها ما يعرف باسم خوارزمية الفقاعة حيث يتم مقارنة كل عنصرين أو مفتاحين متجاورين وتعديل وضعهما حسب الترتيب المطلوب أما النوع الثاني والذي عرف بخوارزميات الاختيار وأشهرها ما يعرف باسم خوارزمية الاختيار المباشر حيث اختيار أكبر المفاتيح أو أصغرها حسب الترتيب المطلوب ووضعها في الترتيب الأول ثم اختيار الأكبر أو الأصغر من بقية المفاتيح ووضعه في الترتيب الثاني وهكذا يتم اختيار العنصر أو المفتاح التالي من بقية المفاتيح إلى أن يبقى مفتاح أو عنصر واحد هو المفتاح الأخير أو العنصر الأخير .

خوارزمية الفقاعة :

١. ادخل عدد المفاتيح أو العناصر (ن) ومصفوفة المفاتيح أو العناصر .
٢. قارن كل عنصر و العنصر الذي يليه مبتدئاً من العنصر الأول مع الثاني، الثاني مع الثالث ... حتى العنصر (ن – ١) ، العنصر (ن) .
٣. إن كانا غير مرتبين الترتيب المطلوب بُدلَ موقعهما (نلاحظ هنا أن المقارنة التالية ستكون بين العنصر السابق والعنصر التالي إذا تمت عملية تبديل الموقعين، مثلاً إذا تم تبديل العنصر الثالث مع الرابع فإن العنصر الرابع الجديد سيكون في الواقع هو العنصر الثالث؛ لذا من الناحية العملية ستتم المقارنة هذه المرة بين العنصر الثالث والعنصر الخامس، وهذا سبب تسمية الفقاعة حيث يقفز العنصر ليعيد الترتيب بسرعة إلى وضعه

الطبيعي مثل الفقاعة الصغيرة الحقيقية تقفز بسرعة إلى سطح الماء أو الزيت تاركة الفقاعات الأكبر داخل السائل) .

٤. قف إذا لم يعد هناك أي تبديل مواقع وإلا عد إلى ٢ .

مثال (٣) :

إذا كان صف المفاتيح هو :

٢٥ ٥٧ ٤٨ ٣٧ ١٢ ٩٢ ٨٦ ٣٣

المطلوب ترتيب العناصر ترتيباً تصاعدياً

مصفوفة العناصر الأساسية :

		٢٥	٥٧	٤٨	٣٧	١٢	٩٢	٨٦	٣٣
التبديل الأول	لا تبديل	٢٥	٥٧						
التبديل الثاني	تبديل	٢٥	٤٨	٥٧					
التبديل الثالث	تبديل	٢٥	٤٨	٣٧	٥٧				
التبديل الرابع	تبديل	٢٥	٤٨	٣٧	١٢	٥٧			
التبديل الخامس	لا تبديل	٢٥	٤٨	٣٧	١٢	٥٧	٩٢		
التبديل السادس	تبديل	٢٥	٤٨	٣٧	١٢	٥٧	٨٦	٩٢	
التبديل السابع	تبديل	٢٥	٤٨	٣٧	١٢	٥٧	٨٦	٣٣	٩٢

مصفوفة العناصر بعد الدورة الأولى :

		٢٥	٤٨	٣٧	١٢	٥٧	٨٦	٣٣	٩٢
التبديل الأول	لا تبديل	٢٥	٤٨						
التبديل الثاني	تبديل	٢٥	٣٧	٤٨					
التبديل الثالث	تبديل	٢٥	٣٧	١٢	٤٨				
التبديل الرابع	لا تبديل	٢٥	٣٧	١٢	٤٨	٥٧			
التبديل الخامس	لا تبديل	٢٥	٣٧	١٢	٤٨	٥٧	٨٦		
التبديل السادس	تبديل	٢٥	٣٧	١٢	٤٨	٥٧	٣٣	٨٦	
التبديل السابع	لا تبديل	٢٥	٣٧	١٢	٤٨	٥٧	٣٣	٨٦	٩٢

مصفوفة العناصر بعد الدورة الثانية :

		٢٥	٣٧	١٢	٤٨	٥٧	٣٣	٨٦	٩٢
التبديل الأول	لا تبديل	٢٥	٣٧						
التبديل الثاني	تبديل	٢٥	١٢	٣٧					
التبديل الثالث	لا تبديل	٢٥	١٢	٣٧	٤٨				
التبديل الرابع	لا تبديل	٢٥	١٢	٣٧	٤٨	٥٧			
التبديل الخامس	تبديل	٢٥	١٢	٣٧	٤٨	٣٣	٥٧		
التبديل السادس	لا تبديل	٢٥	١٢	٣٧	٤٨	٣٣	٥٧	٨٦	
التبديل السابع	لا تبديل	٢٥	١٢	٣٧	٤٨	٣٣	٥٧	٨٦	٩٢

مصفوفة العناصر بعد الدورة الثالثة :

		٢٥	١٢	٣٧	٤٨	٣٣	٥٧	٨٦	٩٢
التبديل الأول	تبديل	١٢	٢٥						
التبديل الثاني	لا تبديل	١٢	٢٥	٣٧					
التبديل الثالث	لا تبديل	١٢	٢٥	٣٧	٤٨				
التبديل الرابع	تبديل	١٢	٢٥	٣٧	٣٣	٤٨			
التبديل الخامس	لا تبديل	١٢	٢٥	٣٧	٣٣	٤٨	٥٧		
التبديل السادس	لا تبديل	١٢	٢٥	٣٧	٣٣	٤٨	٥٧	٨٦	
التبديل السابع	لا تبديل	١٢	٢٥	٣٧	٣٣	٤٨	٥٧	٨٦	٩٢

مصفوفة العناصر بعد الدورة الرابعة :

		١٢	٢٥	٣٧	٣٣	٤٨	٥٧	٨٦	٩٢
التبديل الأول	لا تبديل	١٢	٢٥						
التبديل الثاني	لا تبديل	١٢	٢٥	٣٧					
التبديل الثالث	تبديل	١٢	٢٥	٣٣	٣٧				
التبديل الرابع	لا تبديل	١٢	٢٥	٣٣	٣٧	٤٨			
التبديل الخامس	لا تبديل	١٢	٢٥	٣٣	٣٧	٤٨	٥٧		
التبديل السادس	لا تبديل	١٢	٢٥	٣٣	٣٧	٤٨	٥٧	٨٦	
التبديل السابع	لا تبديل	١٢	٢٥	٣٣	٣٧	٤٨	٥٧	٨٦	٩٢

مصفوفة العناصر بعد الدورة الخامسة :

		١٢	٢٥	٣٣	٣٧	٤٨	٥٧	٨٦	٩٢
التبديل الأول	لا تبديل	١٢	٢٥						
التبديل الثاني	لا تبديل	١٢	٢٥	٣٣					
التبديل الثالث	لا تبديل	١٢	٢٥	٣٣	٣٧				
التبديل الرابع	لا تبديل	١٢	٢٥	٣٣	٣٧	٤٨			
التبديل الخامس	لا تبديل	١٢	٢٥	٣٣	٣٧	٤٨	٥٧		
التبديل السادس	لا تبديل	١٢	٢٥	٣٣	٣٧	٤٨	٥٧	٨٦	
التبديل السابع	لا تبديل	١٢	٢٥	٣٣	٣٧	٤٨	٥٧	٨٦	٩٢

إذن تم الترتيب النهائي .

خوارزمية الاختيار المباشر

نبحث عن أصغر عنصر في المصفوفة، ونقوم بتبديله مع العنصر الأول، ثم نبحث عن أصغر عنصر من بين عناصر المصفوفة من العنصر الثاني الي العنصر الأخير، ونقوم بتبديله مع العنصر الثاني، ثم نبحث عن أصغر عنصر من بين عناصر المصفوفة من العنصر الثالث الي العنصر الأخير، ونقوم بتبديله مع العنصر الثالث، وهكذا الي ان تنتهي كل عناصر المصفوفة وعندها نحصل علي مصفوفة مرتبة.

مثال (٤) :

إذا كان صف المفاتيح هو :

٢٥ ٥٧ ٤٨ ٣٧ ١٢ ٩٢ ٨٦ ٣٣

المطلوب ترتيب العناصر ترتيباً تصاعدياً.

الخطوة ١:

نبحث عن أصغر عنصر من عناصر المصفوفة وهو العنصر رقم [٥] الذي يساوي "١٢" ونقوم بتبديله مع العنصر رقم [١] :

رقم العنصر	[١]	[٢]	[٣]	[٤]	[٥]	[٦]	[٧]	[٨]
	١٢	٥٧	٤٨	٣٧	٢٥	٩٢	٨٦	٣٣

الخطوة ٢:

نبحث عن أصغر عنصر من بين عناصر المصفوفة من العنصر الثاني الي العنصر الأخير وهو العنصر رقم [٥] الذي يساوي "٢٥" ونقوم بتبديله مع العنصر رقم [٢] :

رقم العنصر	[١]	[٢]	[٣]	[٤]	[٥]	[٦]	[٧]	[٨]
	١٢	٢٥	٤٨	٣٧	٥٧	٩٢	٨٦	٣٣

الخطوة ٣:

نبحث عن أصغر عنصر من بين عناصر المصفوفة من العنصر الثالث الي العنصر الأخير وهو العنصر رقم [٨] الذي يساوي "٣٣" ونقوم بتبديله مع العنصر رقم [٣] :

رقم العنصر	[١]	[٢]	[٣]	[٤]	[٥]	[٦]	[٧]	[٨]
	١٢	٢٥	٣٣	٣٧	٥٧	٩٢	٨٦	٤٨

الخطوة ٤:

نبحث عن أصغر عنصر من بين عناصر المصفوفة من العنصر الرابع الي العنصر الأخير وهو العنصر رقم [٤] الذي يساوي "٣٧" ونقوم بتبديله مع العنصر رقم [٤] وهو نفس العنصر:

رقم العنصر	[١]	[٢]	[٣]	[٤]	[٥]	[٦]	[٧]	[٨]
	١٢	٢٥	٣٣	٣٧	٥٧	٩٢	٨٦	٤٨

الخطوة ٥:

نبحث عن أصغر عنصر من بين عناصر المصفوفة من العنصر الخامس الي العنصر الأخير وهو العنصر رقم [٨] الذي يساوي "٤٨" ونقوم بتبديله مع العنصر رقم [٥] :

رقم العنصر	[١]	[٢]	[٣]	[٤]	[٥]	[٦]	[٧]	[٨]
	١٢	٢٥	٣٣	٣٧	٤٨	٩٢	٨٦	٥٧

الخطوة ٦:

نبحث عن أصغر عنصر من بين عناصر المصفوفة من العنصر السادس الي العنصر الأخير وهو العنصر رقم [٨] الذي يساوي "٥٧" ونقوم بتبديله مع العنصر رقم [٦] :

رقم العنصر	[١]	[٢]	[٣]	[٤]	[٥]	[٦]	[٧]	[٨]
	١٢	٢٥	٣٣	٣٧	٤٨	٥٧	٨٦	٩٢

الخطوة ٧:

نبحث عن أصغر عنصر من بين عناصر المصفوفة من العنصر السابع الي العنصر الأخير وهو العنصر رقم [٧] الذي يساوي "٨٦" ونقوم بتبديله مع العنصر رقم [٧] وهو نفس العنصر:

رقم العنصر	[١]	[٢]	[٣]	[٤]	[٥]	[٦]	[٧]	[٨]
	١٢	٢٥	٣٣	٣٧	٤٨	٥٧	٨٦	٩٢

ونجد العنصر الأخير يكون في موضعه الصحيح.

إذن تم الترتيب النهائي .

الخوارزمية

١. ضع رقم العنصر =١.
٢. ابحث عن أصغر عنصر من عناصر المصفوفه من رقم العنصر الي اخر عنصر.
٣. قم بتبديل العنصر الأصغر مع العنصر الذي يشير اليه رقم العنصر.
٤. رقم العنصر =رقم العنصر +١.
٥. إذا كانت رقم العنصر = اخر عنصر اذهب الي ٦ وإلا اذهب الي ٢.
٦. اطبع المصفوفة وهي مرتبة.
٧. النهاية.

برنامج الفقاعة

```
Program bubble_sort ;
Const
     n: = 50;
Var
   I, Pass : integer;
   Temp = integer;
   Sortfield = array [1..n] of integer;
   Exchange: Boolean;
Begin
   for I: = 1 to n do
        readln (sortfield[I]);
   Pass : = 0 ;
   exchange: = true;
   While exchange do
   begin
     Exchange : = flase;
     pass: = pass + 1;
     For I : = to n-1 do
       If sortfield[I]>sortfiled[I+1] then
       begin
           temp: = sortfiled [I] ;
           sortfield[I] := sortfiled[I +1];
           sortfiled[I +1]: = temp;
           Exchange : = true;
       end ;
   end ;
   Writeln(' no of passes' pass);
   For I : = I to  n do
          writeln (sortfiled[I]);
End.
```

برنامج الفقاعة

```
برنامج خوارزمية الفقاعة
ثابت
    ن = ٥٠ عدد العناصر
المتغيرات
    دورة ومؤشر  : عددصحيح
    مؤقت : عدد صحيح
    قائمة : مصفوفة من ١ الي ن وهي رقمية
    تبديل : منطقي .
أبدأ
    المؤشر من  ١ إلى ن
        أقرأ عناصر مصفوفة القائمة
    دورة = ٠
    تبديل = نعم
    ما دام التبديل نعم  أعمل
        أبدأ
        التبديل = لا .
        الدورة = الدورة + ١
        من ١ إلى ن -١  افعل
        إذا كان بيان حقل التبديل < من الذي يليه
        أبدأ
            أجعل حقل التبديل= تخزين مؤقت ،
            أجعل حقل التبديل التالي=حقل التبديل
            أجعل  النخزين المؤقت =التبديل التالي .
            أجعل التبديل = نعم
        نهاية (* نهاية اذا كان *)
    نهاية (* ما دام *)
    اكتب "عدد دورة الترتيب " ، دورة
    من ١ إلى ن
        اكتب  البيانات المرتبة من القائمة
نهاية البرنامج
```

٥. خوارزميات تشفير المعلومات :

كلمة تشفير نابعة من كلمة سايفر **(cipher)** الإنجليزية والتي هي في الواقع أصلها الكلمة العربية صفر والتي تعني (لا شيء) إذ أن التشفير مقصود به ألا يفهم القارئ شيئاً مما هو مكتوب (غير ذلك الشخص المصرح له بذلك). وبهذا لابد أن يكون علم التشفير علماً قديماً ؛ فالناس منذ الأزل لهم أسرارهم التي لا يودّون أن يعرفها كل الناس. ومما هو مسجل في التاريخ أن المصريين القدامى أول من بدأ التشفير بطريقة علمية، وذلك لما لديهم من مهارات رياضية معروفة, ثم مارس الهنود والإغريق وغيرهم من الأمم القديمة التشفير، و أن المسلمين والعرب مثل ابن الدرهم قد وثقوا لعلم التشفير وكتبوا فيه ، أما الألمان فهم أول من مارس التشفير باستخدام الآلة حيث استخدموا الآلة المعروفة بـــ **(Enigma)** .

تتبع نظم التشفير الخطوات التالية :

١. صمم خوارزمية الشفرة التي تود استخدامها مع الطرف الآخر .
٢. ادخل النص أو المعلومات واضحة .
٣. ادخل الشفرة على النص أو المعلومات ليخرج نص أو معلومات مشفرة .
٤. ارسل الرسالة إلى الطرف الآخر .
٥. عالج النص أو المعلومات التي بالرسالة بخوارزمية فك الشفرة .
٦. اخرج النص واضحاً أو المعلومات واضحة .

شفرات الاستبدال أو التعويض :

تعتبر خوارزميات الاستبدال والتعويض من الخوارزميات القديمة أبسطها هي خوارزمية يوليوس قيصر والتي تقوم باستبدال أي حرف بالحرف الثالث في الترتيب . يعني مثلاً حسب ترتيب الحروف العربية بالنظام التالي:

١٤	١٣	١٢	١١	١٠	٩	٨	٧	٦	٥	٤	3	2	1
ص	ش	س	ز	ر	ذ	د	خ	ح	ج	ث	ت	ب	أ

٢٨	٢٧	٢٦	٢٥	٢٤	٢٣	٢٢	٢١	٢٠	١٩	١٨	١٧	١٦	١٥
ي	و	هـــ	ن	م	ل	ك	ق	ف	غ	ع	ظ	ط	ض

فإن أ تصبح ث وب تصبح ج ول تصبح هـــ ون تصبح ي وهكذا .إذن نضيف لموقع الحرف المراد تشفيره ٣ ليصبح الحرف المشفر هو حرف ذلك الموقع .

مثال (٥) :

حسب خوارزمية يوليوس قيصر ما التشفير المقابل للحروف "هـــ" ، "و" ،"ي" مستعيناً بالجدول أعلاه؟

- موقع الحرف "هـــ" =٢٦ ، موقع الحرف بعد التشفير =٢٦+٣=٢٩ لا يوجد حرف رقمه ٢٩ إذن نقوم بطرح ٢٨ وهي عدد الحروف وعليه يكون الحرف المقابل للحرف "هـــ" هو الحرف الذي موقعه =٢٩-٢٨=١ أي الحرف "أ"
- موقع الحرف "و" =٢٧ ، موقع الحرف بعد التشفير =٢٧+٣=٣٠-٢٨=٢ وهو الحرف "ب"
- موقع الحرف "ي" =٢٨ ، موقع الحرف بعد التشفير =٢٨+٣=٣١-٢٨=٣ وهو الحرف "ت"

هذا الاجراء يتم في الحاسوب تلقائياً بواسطة الدالة **MOD**.

معلومة

تحسب الدالة MOD باقي القسمة فمثلاً

- 16 MOD 3 =1 , 37 MOD 7 =2
- 89 MOD 11=1 , 30 MOD 28=2
- 31 MOD 28=3 , 7 MOD 4 =3

وإذا أردنا تنفيذ هذه الخوارزمية بالحاسوب بلغة باسكال فإننا ببساطة .

```
Program  Substitution;
Const
     No_of_char = 28;
Var
  I: integer;
  letters: array[1..no_of_char]of char;
begin
  for I: = 1 to no_of_char do
    letters[I]:=char((i+3)mod No_of_char);
End
```

برنامج استبدال الأحرف

ثابت

عدد الأحرف = ٢٨

متغيرات

الأحرف = مصفوفة حروف من ١ إلى ٢٨

أبدأ

من ١ إلى عدد الأحرف

اجعل حرف الاستبدال = الحرف الذي بعد ثلاثة مواقع منه وعند انتهاء الحروف أبدأ من الأول .

النهاية

أما إذا أردنا فك الشفرة فإننا نكرر نفس البرنامج فقط نجعل المعادلة

```
letters[I]:=char((I-3)mod No_of_char)
```

أي أن حرف المستبدل هو الحرف السابق قبل ثلاث مواقع من حرف الشفرة وعندما تكون القيمة سالبة فإننا نضيف "٢٨" وهي عدد الحروف .

مثال (٦) :

حسب خوارزمية يوليوس قيصر ما الحرف الأصلي للمقابل للحروف المشفرة "أ" ، "ب" ،"ت" مستعيناً بالجدول السابق.

- موقع الحرف المشفر "أ" =١ ، موقع الحرف المقابل =١-٣=-٢ لا يوجد حرف رقمه -٢ إذن نقوم بجمع ٢٨ وهي عدد الحروف وعليه يكون الحرف المقابل للحرف "أ" هو الحرف الذي موقعه =-٢+٢٨=٢٦ أي الحرف "هـ".
- موقع الحرف المشفر "ب" =٢ ، موقع الحرف المقابل =٢-٣=-١+٢٨=٢٧ وهو الحرف "و".
- موقع الحرف المشفر "ت" =٣ ، موقع الحرف المقابل =٣-٣=٠+٢٨=٢٨ وهو الحرف "ي".

مثال (٧) :

حسب خوارزمية يوليوس قيصر ما تشفير عبارة "أوقف الشراء" مستعيناً بالجدول أعلاه.

سيكون الألف بعد الشفرة ثاء والواو "ب" والقاف ميماً والفاء لاماً واللام تصبح ياء والشين تصبح طاء والراء تصبح شيناً ومن ثم تصبح العبارة (اوقف الشراء) (ثنمل ثهطشث)) وإذا عرف الطرف الآخر مفتاح الاستبدال ٣ فإنه سيفك تلك الشفرة ويصل إلى النص الواضح .

فك الشفرات

تعتمد خوارزميات فك الشفرات علي الأتي:

١. المعرفة بخواص اللغة مثل تكرارية الحروف في اللغة، ففي كل لغة تتكرر الحروف بنسب ثابتة، وهناك إحصائية ثابتة للنسبة المئوية لتكرار أي حرف في أي لغة، ثم أن هناك حروفاً معينة تتكرر في الكلمة الواحدة وحروف أخرى لا تتكرر مثل حروف العطف في اللغة العربية فيمكن أن تجد الواو جوارها واواً أو الفاء جوارها فاءً وكذلك حروف الجر فيمكن أن تجد جوار اللام لاماً وجوار الباء باءً ولكن قل أو لا تكاد تجد جوار القاف قافاً أو جوار النون نوناً . كذلك هناك حروف يكثر تجاورها مثل ألف ولام في اللغة العربية وكيو يو . q u في اللغة الإنجليزية .

كذلك هناك حروف تكثر في بداية الكلمات في كل لغة مثل حرف الألف في اللغة العربية وحرف الآي I في اللغة الإنجليزية .

٢. معرفة أسلوب المنافس اللغوي والمنطقي والعلمي وذلك عن طريق كتاباته المفتوحة و كلامه وحواره مع الآخرين .

٣. تخمين نوع الرسائل التي يمكن أن يرسلها بناءً على معلومات عن أنشطته وعلاقاته مع الطرف الآخر ، فهل هي رسائل تجارية ، رياضية ، أمنية ، ثقافية ، اكاديمية.

٤. محاولة الوصول بقدر الإمكان إلى أنواع الخوارزميات التي تستخدم في التشفير فإذ عرفنا على سبيل المثال أنه يستخدم شفرة قيصر يمكن معرفة المفتاح بسهولة عن طريق خواص حروف اللغة ثم تجربة استبدال حرف

بآخر حتى نصل للحل .لابد من الإشارة في أن سرعة الوصول إلى فك الشفرة يتناسب تناسباً طردياً وأسياً مع جزيئات المعلومة المكتشفة. مثلاً إذا اكتشفنا الحرف ألف يمكن أن نكتشف الحروف لام والحرف ن وكل الحروف التي يكثر تجاورها للألف، ومن ثم كل حرف يتم اكتشافه يؤدي إلى اكتشاف الحروف الأخرى وهكذا تتزايد سرعة اكتشاف الحروف أضعافاً مضاعفة، وهذا ما يعرف في علم تعقيدات الحاسوب بالسرعة الأسية .

تعقيدات شفرات التبديل والتعويض :

يمكن تعقيد شفرة يوليوس قيصر ليكون مفتاحها بدلاً من الرقم ثلاثة أي رقم اخر مثلاً ان يكون المفتاح "٧" فإن "أ" ستصبح بعد عملية التشفير "د" ، "ن" ستصبح بعد عملية التشفير "ث". وهكذا

نلاحظ في شفرة يوليوس قيصر أن كل الحروف يتم تشفيرها بمفتاح واحد فقط فإذا اكتشف ذلك المفتاح سيتم فك الشفرة .

يمكن تعقيد هذه الشفرة بإنشاء جدول حروف يكون لكل حرف مفتاح خاص يتم تحديد هذا المفتاح عشوائياً (يختلف هذا الجدول اختلافاً كلياً عن جدول ترتيب الحروف) ويكون هذا الجدول الجديد مفتاح الشفرة السري بين الراسل والمرسل إليه. ويتم هذا الإجراء في الحاسوب تلقائياً بواسطة الدالة **RANDOM** .

معلومة

الداله **RANDOM (N)** تعطي رقماً عشوائياً بين ١ و N فمثلاً:

- يمكن ان تعطي١، ٢، ٣، ٤، ٥، ٦ **RANDOM (6)**
- يمكن ان تعطي أي رقم بين ١و٢٨ **RANDOM (28)**

وحتى يتم ذلك يمكن عمل تعديل بسيط في البرنامج السابق وذلك باستدعاء الدالة العشوائية الرقمية أي تلك التي تعطي رقماً عشوائياً بين واحد وعدد الحروف حسب نظام التمثيل واستخدام مصفوفة منطقية لتحدد هل الرقم العشوائي تم استخدامه أم لا على النحو التالي :

```
Program random_change;
Const
   n =  28;
Var
   L = integer;
   Substit : array [1..n] of integer;
   Character: array [1.. n] of char;
   Change: array[1.. n] of Boolean;
Begin
    For I: = 1 to n do
    Begin
        Substit[I]: = I;
        Change[I]: = flase;
    End
    For I : = 1 to n do
    Begin
       L: = random(n);
       If Not change [L] then
       Begin
          Substit[I] : = L;
         Change[I]: = true;
       End;
       Characters[I]: = char(substit[I]) ;
    End;
End.
```

```
برنامج تبديل الحروف
ثابت
    عدد الحروف ٢٨
متغيرات
    م  :  عدد صحيح
    مصفوفة تغيير الأرقام ،
    مصفوفة الحروف المشفرة
    مصفوفة هل استخدم الرقم العشوائي ؟
أبدأ
    البرنامج من ١ إلى عدد الحروف ،
    أبدأ
        اجعل كل رقم في مكانه  ،
        اجعل هل تم التغيير = لا لكل الحروف
    النهاية
    من ١ إلى عدد الحروف
    أبدأ
        اجعل م= الرقم العشوائي الذي تم توليده من ١ الي  ن
        هل تم تغيير في الحروف رقم م من قبل
        أبدأ  ،
            وضع رقم الحرف إلى الرقم العشوائي
            اجعل هل تم التغيير = نعم
        النهاية
        الحرف المشفر في المصفوفة =الحرف المقابل لرقم الحرف
    النهاية
انتهت عملية تغيير وضع الأرقام
```

هذا البرنامج يقوم أولاً بافتراض كل الحروف في وضعها الطبيعي، ثم بعد ذلك يتم توليد أرقاماً عشوائية بالدالة **random** ولكن لأن هناك احتمالاً بأن تعطي الدالة العشوائية أرقاماً متكررة ينتج عنها حرفان أو أكثر في موقع واحد فكان لابد من استخدام المصفوفة المنطقية لإزالة التكرارية، ولابد من إعطاء القيم الأصلية للحروف لمصفوفة التبديل حتى تظل الأرقام التي لم يحدث لها تغيير في وضعها القديم .

إن نقطة الضعف في هذه الشفرة تتمثل في أنها يمكن أن تكتشف باستخدام التكرار النسبي للحروف في اللغة كما ذكرنا من قبل؛ لأن كل حرف يقابله حرف وبهذا ستظل التكرارية النسبية للحروف كما هي مع تغيير الحرف ومن ثم يمكن اكتشاف الحرف المقابل وفك الشفره.

ملحوظة :

لاحظ أننا لم نعط المسافة بين الكلمات رمزاً في الأمثلة من أجل المثال ولكنها في الواقع هي مثلها مثل الحروف لها رمز في الحاسوب ويسري عليها التبديل بكل أنواعه مثلها مثل الحروف .

٦. ترميز هوفمان :

عندما تحدثنا عن أنظمة الترميز الثنائي في منهج السنة الأولى ذكرنا أن كل حرف لابد أن يكون به رمز ثنائي مفرد ، ومن هنا تطورت أنظمة الترميز الثنائي من نظام بي سي دي (**B C D**) السداسي إلى نظام البسيديك (**EBEDIC**) ونظام آسكي (**ASCII**) الثماني . ولكن ذكرنا في الفقرة السابقة أن لكل لغة خواصها في التكرار النسبي للحروف وداخل اللغة لكل مؤسسة أو مركز معلومات خواص في نوع المعلومات التي يتعامل معها،ا فإن كان مركزاً إحصائياً فإن أكثر المعلومات

ستكون أرقاماً وإن كان مركزاً صحفياً فإن أكثر المعلومات ستكون صوراً وأما الراديو فإن أكثر المعلومات كلاماً .

من هذا المنطلق كان لابد من تحوير التمثيل الثنائي أو تغييره ليمكن من تمثيله تمثيلاً أمثل للبيانات بناء على خواص اللغة وخواص البيانات، ونعني بالتمثيل الأمثل هو التمثيل الذي يأخذ أقل حجم ممكن من ذاكرة الحاسوب، أو وسائط تخزين الحاسوب. ومن ثم أقل وقت ممكن في إرسال تلك البيانات وهذا يتم ببساطة بتمثيل أكثر الحروف أو الأرقام أو الرموز تكراراً بأقل عدد من الثنائيات، ويزيد عدد الثنائيات الممثلة للحرف أو الرقم أو الرمز كلما قلت تكرارية استخدامه . وهذه الطريقة أو الخوارزمية تعرف بخوارزمية هوفمان ويتم تصميمها بطريقة الشجرية الثنائية على النحو التالي :

١- نفترض أن لدينا خمس رموز أ١ و أ٢ وأ٣ وأ٤ وأ٥ وكان التكرار النسبي لهذه الرموز مرتبة تنازلياً حسب الجدول أدناه :

الرمز	التكرار النسبي
أ١	٤٠٪
أ٢	٢٠٪
أ٣	٢٠٪
أ٤	١٠٪
أ٥	١٠٪

٢- يتم ربط كلّ رمزين من أسفل إلى أعلى بالترتيب التالي :

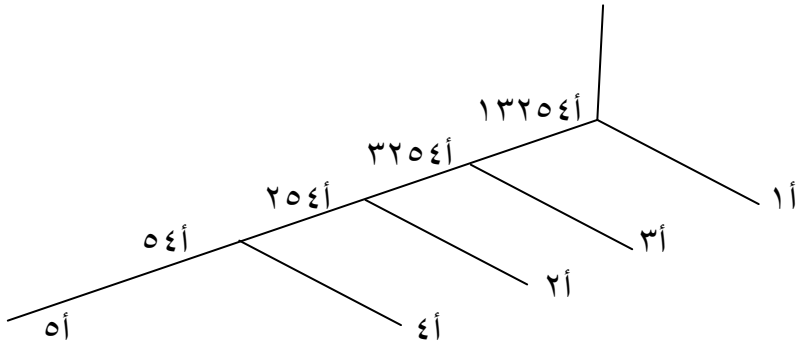
أ) أ٤ و أ٥ يتم ربطهما ويستبدلا برمز ربط جديد اسمه أ٥٤ وبتكرارية افتراضية تساوي جمعهما النسبي ١٠٪ + ١٠٪ = ٢٠٪ .

ب) الآن لدينا ٤ رموز هم الرمز أ١ بتكرار نسبي ٤٠٪ والرموز أ٢ وأ٣ وأ٥٤ بتكرار نسبي ٢٠٪ فيمكننا اختيار أي اثنين من أ٢ وأ٣ و أ٥٤ فمثلاً أ٢ و أ٥٤ وربطها برمز ربط جديد سمه أ٢٥٤ بتكرار نسبي ٤٠٪ .

ج) الآن أصبح لدينا ثلاث رموز هم أ١ وأ٣ و أ٢٥٤ بتكرار نسبي ٤٠٪ و٢٠٪ و٤٠٪ على التوالي . لذا يتم ربط أ٣ مع أي من أ١ أو أ٢٥٤ فمثلاً أ٢٥٤ برمز ربط جديد سمه أ٣٢٥٤ بتكرار نسبي ٦٠٪ .

د) الآن بقي لنا رمزين هم أ١ و أ٣٢٥٤ يتم ربطهما برمز جديد اسمه أ١٣٢٥٤ بتكرار نسبي ١٠٠٪ .

بهذا تكتمل شجرية هوفمان حسب الرسم أدناه .



نلاحظ أن الشجرية مائلة بل شبه ساقطة على الجانب الأيسر أما تكويد الرموز فحسب الجدول أدناه .

الرمز	التكويد
أ١	١
أ٢	١٠٠
أ٣	١٠
أ٤	١٠٠٠
أ٥	١٠٠٠

متوسط حجم الشجرية :

يقاس متوسط حجم الشجرية بطول الرمز مضروب في التكرارية بالنسبة للرمز فمن جدول التكرار النسبي وجدول التكويد يمكن حساب حجم هذه الشجرية بـــ ١×٤٠٪ + ٣×٢٠٪ + ٢×٢٠٪ + ٤×١٠٪ + ٤×١٠٪ = ٢,٢ ثنائية / الرمز. تعتبر من ميزات تكويد هوفمان أن يعطي كما ذكرنا التكويد الأمثل، ومن ثم حتى إذا تم تغبير اختيار ربط الرموز التي لها في كل مرحلة نفس التكرار النسبي لن يؤثر في القيمة النهائية لمتوسط حجم الشجرية .

إن الرموز أ١ وأ٢ وأ٣ وأ٤ وأ٥ يمكن أن تكون رموزاً لآي رسائل أو لكلمات أو توجيهات هامة، ولها تكرارية مثلاً يمكن أن تكون رسائل بين مكتبين في بورتسودان والخرطوم في شركة استيراد، وبها تكرارية سلم البضاعة للزبون ٤٠٪ وتستلم البضاعة من الباخرة ٢٠٪ وخزن البضاعة ٢٠٪ وإرسال البضاعة إلي الخرطوم ١٠٪ وتسلم التحويل المالي ١٠٪ أو يمكن أن تكون نفس هذه الرموز يرمز بها لرسائل عسكرية في العمليات مثل (تسلّم الأغذية) ٤٠٪ (وتسلّم الذخيرة) ٢٠٪ (وتسلّم الرواتب) ٢٠٪ و(الهجوم على العدو) ١٠٪ و(إيقاف الهجوم) ١٠٪ .

تكويد هوفمان المثالي للحروف الإنجليزية :

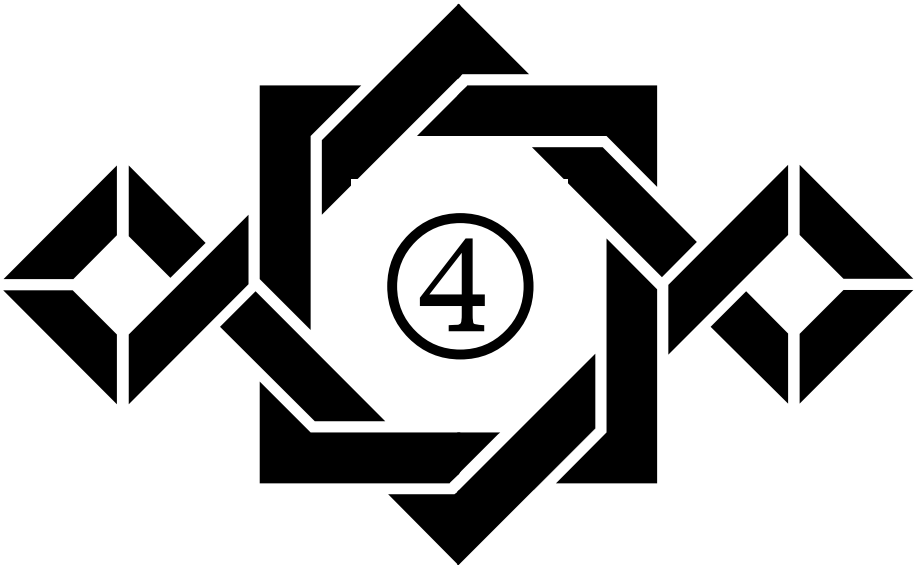
إذا كانت تكرارية الحروف الإنجليزية حسب دراسات إحصائية مكثفة للغة الإنجليزية كانت حسب الجدول المرفق- فإن خوارزمية هوفمان يمكن أن تصمم على النحو التالي :

١. اربط z و q في qz بنسبة ٠,٥٪ .
٢. اربط qz مع x بنسبة ١٪ من xqz .
٣. اربط k و j في jk بنسبة ١٪ .
٤. اربط xqz و jk في jk xqz بنسبة ٢٪ .

٥. اربط g و v في vg بنسبة ٢,٥٪ .
٦. اربط b و w في wb بنسبة ٣٪ .
٧. اربط p و y في yp بنسبة ٤٪ .
٨. اربط f مع jkxqz في jkxqzf بنسبة ٤٪ .
٩. اربط vg و wb في wbvg بنسبة ٥,٥٪ .
١٠. اربط m و u في um بنسبة ٦٪ .
١١. اربط c و l في lc بنسبة ٦,٥٪ .
١٢. اربط d و yp في ypd بنسبة ٨٪ .
١٣. اربط jkxqzf مع wbvg في wbvgjkxqzf بنسبة ٩,٥٪ .
١٤. اربط s و h في sh بنسبة ١٢٪ .
١٥. اربط um مع j في jum بنسبة ١٢,٥٪ .
١٦. اربط c مع r في rc بنسبة ١٣٪ .
١٧. اربط o مع n في on بنسبة ١٥٪ .
١٨. اربط a مع ypd في aypd بنسبة ١٦٪ .
١٩. اربط wbvgdkxqzf مع t في twbvgdkxqzf بنسبة ١٨,٥٪ .
٢٠. اربط e مع sh في esh بنسبة ٢٥٪ .
٢١. اربط jum مع vlc في vlcjum بنسبة ٢٥,٥٪ .
٢٢. اربط on مع apyd في apydon بنسبة تكرارية ٣١٪ .
٢٣. اربط twbvghjxqzf مع esh في eshtwbvgjkxqzf بنسبة ٤٣,٥٪ .
٢٤. اربط rkcium مع apydib في apydibrkcium بنسبة تكرارية ٥٦,٥٪.
٢٥. اربط eshtwbvgkxqzf مع apydibrkcium في apydonrlcjumeshtwbvgjkxqzf بنسبة تكرارية ١٠٠٪ .

٧. تمرين

١. ما الغرض من معرفة أماكن تخزين المعلومات ؟
٢. ما المقصود بالبحث المتتالي؟
٣. ما المقصود بالبحث الثنائي ؟
٤. ما المفتاح الأساسي؟
٥. حدد عيوب استخدام الأسماء كمفاتيح مع ضرب أمثله لذلك.
٦. أيهما افضل للبحث المتتالي أم الثنائي ؟ وضح الأسباب.
٧. صف خوارزمية البحث المتتالي.
٨. صف خوارزمية البحث الثنائي.
٩. ما المقصود بتصنيف المعلومات ؟
١٠. صف أحد خوارزميات التبديل.
١١. صف أحد خوارزميات الاختيار.
١٢. أيهما افضل للاستخدام في ترتيب قائمة من الأعداد اسلوب الفقاعة، أم اسلوب الاختيار المباشر ؟ وضح الأسباب.
١٣. عرف الشفرة .
١٤. ما الغرض من استخدام الشفرة ؟
١٥. حدد خطوات التشفير.
١٦. علي ماذا يعتمد فك الشفرة .
١٧. اعمل شفرة شبيه بشفرة يوليوس قيصر مستخدما الرقم "٥" كمفتاح لهذه الشفرة.
١٨. وضح عيب الشفرة التي عملتها في المثال السابق.
١٩. ما الحل المناسب حسب رأيك لتفادي عيب الشفرة في المثال السابق ؟

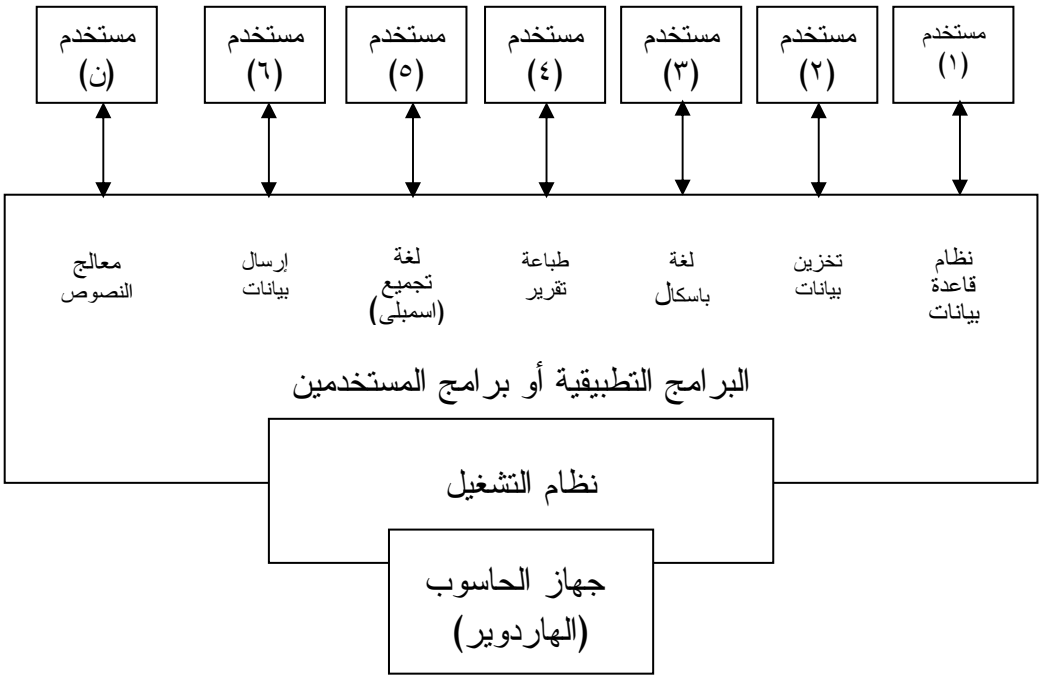


نُظُم التّشغيل

نُظُم التّشغيل

١. مدخل :

نظام التشغيل هو برنامج يعمل بين مستخدم الحاسوب وجهاز الحاسوب ليمكن المستخدم من تنفيذ برمجياته بسهولة وفعالية وبكفاءة عالية (نعني بالكفاءة العالية التنفيذ في أقل وقت ممكن وفي أقل مساحة من الذاكرة) يمكن أن نتصور نظام التشغيل كتنظيم فرعي من نظام الحاسوب في الرسم التالي :



نلاحظ من الرسم أن نظام التشغيل ينظم أو يتحكم في كل طلبات المستخدمين، فمنهم من يريد استخدام برنامج قاعدة البيانات، ومنهم من يريد استخدام برنامج تخزين ملفات على القرص، ومنهم من يريد تشغيل برنامج بلغة باسكال لذا فهو يطلب من نظام التشغيل مترجم باسكال، ومنهم من يريد طباعة

تقرير أو طباعة رسم بياني، ومنهم من يريد استخدام لغة التجميع، ومنهم من يريد استفساراً أو تعديل بيانات على الشاشة، ومنهم من يريد إرسال بيانات عبر شبكة الاتصالات، ومنهم من يريد استخدام برنامج معالج النصوص إلى آخر طلبات المستخدمين التي يؤديها الحاسوب. وأحياناً بل في كثير من الأحيان قد يطلب عدد من المستخدمين نفس الخدمة في وقت واحد . إذن يمكن أن نقول أن برنامج نظام التشغيل هو برنامج تحكم يساعد المستخدمين في تنفيذ برامجهم وإنهاء أعمالهم باستخدام الحد الأدنى من إمكانات الحاسوب (الوقت والذاكرة) .

٣. ما قبل نظم التشغيل :

عند اختراع الحاسوب في نهاية الأربعينات كان التعامل مع الحاسوب باللغة الثنائية أو ما يعرف بلغة الماكينة، وكان مستخدم الحاسوب لابد أن يكون مبرمجاً بلغة الماكينة ولم يكن حينذاك يوجد نظام تشغيل أو مشغل؛ بل كان المبرمج نفسه الذي يشغل الحاسوب ليقوم بتنفيذ برنامجه، ثم بعد ذلك يدخل المبرمج التالي ويشغل الحاسوب لتنفيذ عمله وهكذا. ولتنظيم أعمال المبرمجين يقوم المبرمج بحجز الحاسوب لفترة معينة كل ساعة، ثم الذي يليه فإذا اكتملت الساعة ولم ينفذ المبرمج عمله يوقف عمله بتسليم الحاسوب للمبرمج التالي . تم في نهاية الخمسينيات تصميم المترجمات مثل مترجم ألقول (باسكال حالياً)، ومترجم فورتران، ومترجم كوبول. وقد سهلت المترجمات عملية تصميم البرامج وساعدت في كتابة برامج متطورة ومتعددة ولكن لم تساعد في عملية التشغيل، وظل التشغيل يأخذ وقتاً طويلاً حيث يتم أولاً انزال المترجم من الشريط إلى الذاكرة، ثم بعد ذلك انزال البرنامج ثم تشغيل المترجم على البرنامج ثم تشغيل المترجم على البرنامج ليتحول إلى لغة التجميع ثم إعادة المترجم إلى الشريط على البرنامج المترجم إلى لغة التجميع للتصحيح باللغة الثنائية ، ثم إعادة لغة التجميع إلى الشريط ثم تشغيل البرنامج المترجم باللغة الثنائية

أو ما يعرف بالبرنامج الهدف على الحاسوب . وإذا حدث أي خطأ في البرنامج الأساسي يتم تصحيح الخطأ وإعادة كل هذه العمليات من جديد . هذا من ناحية ، من ناحية أخرى خلال كل فترة إنزال شريط لغة البرمجة وإعادتها للشريط وإنزال شريط لغة التجميع وإعادتها إلى الشريط يظل الحاسوب عاطلاً غير مستخدم فإذا علمنا أن تكلفة ساعة الحاسوب في تلك الفترة حوالي ٤٥ دولار في الساعة إذا عمل لمدة ٢٤ ساعة وحوالي ٩٠ دولار في الساعة إذا عمل لفترة ١٢ ساعة في حين أن الحد الأدنى للأجور في تلك الفترة كان فقط دولاراً واحداً للساعة بالولايات المتحدة . ممكن أن نرى مدى الخسارة الناتجة عن أي وقت يضيع دون استخدام الحاسوب لتنفيذ أعمال مفيدة . لهذا تم خلق وظيفة جديدة في الحاسوب هي وظيفة المشغل وهو الذي يقوم بأداء كل الأعمال اليدوية من إنزال للأشرطة وغيرها والتي كان يقوم بها المبرمجون، وقد أدى ذلك إلى تقليل الزمن الضائع من الحاسوب لما كان للمشغلين من خبرة في هذه الأعمال اليدوية أكثر من المبرمجين ثم أنه عندما يظهر خطأ في البرنامج – فإن المشغل يقوم مباشرة بإيقاف ذلك البرنامج وإعادته للمبرمج والبدء في تشغيل البرنامج التالي. ولقد كان في السابق كما ذكرنا [عندما لم يكن هناك مشغل يقوم المبرمج باستخدام كل وقته ويحاول تصحيح برنامجه إن كانت به أخطاء خلال وقته المحجوز] ولكن بعد خلق وظيفة المشغل يتم الاستفادة من الوقت الذي يصحح فيه المبرمج برنامجه في تشغيل برنامج آخر . كذلك لتقليل الوقت الضائع في إنزال وإعادة أشرطة المترجمات والتجميع أدخلت فكرة حزم الأعمال وهي أن يتم تشغيل كل مجموعة أعمال متساوية في وقت واحد مثلاً كل البرامج المستخدمة للغة "ألقول" يتم تشغيلها مع بعضها حيث يتم إنزال مترجم "ألقول" لكل هذه الأعمال في وقت واحد . وبالمثل برامج فورتران أو برامج كوبول وبهذا يتم توفير وقت إنزال و إعادة المترجم مع أي عمل مفرد . هذه الطريقة بالتأكيد وفرت وقتاً كبيراً، ولكنها تتطلب من المشغل أن يكون مراقباً للحاسوب

مراقبة لصيقة حتى إذا حدث خطأ في أحد برامج الخدمة يقوم المشغل بتوجيه الحاسوب للانتقال للبرنامج التالي ويظل الحاسوب عاطلاً خلال تلك الفترة أما إذا غفل عن ذلك سيظل الحاسوب عاطلاً حتى ينتبه [وبهذا يظل وقتاً ضائعاً في الانتقال من برنامج إلى آخر داخل الحزمة إذا حدث خطأ، وهناك حاجة لمراقبة لصيقة من المشغل] . لمعالجة هذه المشكلة تم إدخال فكرة التوالي التلقائي للأعمال وهي عبارة عن برنامج صغير يقوم بتعريف الحاسوب بالأعمال، أو البرامج المطلوبة على التوالي. فإذا تم إكمال البرنامج أو حدث به خطأ ينتقل تلقائياً للبرنامج التالي دون الحاجة لتدخل المشغل، وهذه الفكرة أو هذا البرنامج والذي يسمى بالمراقب المقيم resident monitor يعتبر أول خطوة في تصميم برنامج نظام التشغيل .

٣. المراقب المقيم:

مثال لأوامر برنامج المراقب المقيم :

تشغيل مترجم فورتران	$ F T N
تشغيل لغة التجميع اسمبلا	$ A S M
تشغيل برنامج المستخدم	$ R U N

وللتفريق بين عمل وآخر لابد أن يكون :

أول سطر في العمل	$ JOB
وآخر سطر في العمل	$ END

وهذان السطران يمكن عن طريقهما معرفة الزمن الذي استخدمه كل عمل. كذلك متاح إضافة أسطر أخرى فيها مثلاً اسم العمل أو البرنامج المراد تشغيله ورقم المستخدم حتى تحسب جملة الزمن الذي قضاه في كل برامجه لحساب المبلغ المطلوب منه . نلاحظ من الأسطر أعلاه أنها تبدأ بعلامة دولار ($) وفي بعض

الأجهزة مثل (IBM) كانت تبدأ بعلامة (//) وهذه العلامة ضرورية للتفريق بين أسطر أوامر التشغيل وأسطر البرنامج والمدخلات وعليه سيتم تشغيل حزمة الأعمال على النحو التالي :

بداية عمل جديد	$ JOB
تشغيل مترجم فورتران	$ FTN
البرنامج	
ترجمة برنامج فورتران إلى لغة الماكينة	$ LOAD
تشغيل البرنامج على البيانات	$ RUN
البيانات	
نهاية العمل	$ END
بداية عمل جديد	$ JOB

٤. المراقب والمستخدم:

لقد أحدث برنامج المراقب المقيم مع حزمة الأعمال تطوراً كبيراً في رفع كفاءة تشغيل الحاسوب، وقد رأينا أن برنامج المراقب المقيم يقوم بتشغيل الأعمال داخل الحزمة عملاً بعد عمل على التوالي، ولكن هناك مشكلة إذا كان هناك خطأ في برنامج أحد أعمال الحزمة مثل كتابة أمر غير موجود في لغة البرمجة أو محاولة إدخال بيانات في مواقع غير معرفة في الذاكرة (مثلاً أن تكون المصفوفة أ معرفة من ١ إلى ٢٥ ثم محاولة إدخال بيانات في الموقع أ (٢٦) أو أ (صفر) ونقول في هذه الحالة أن برنامج المراقب المقيم قد وقع في مصيدة وسيضطر إلى مسح ذلك البرنامج من الذاكرة ثم البدء في العمل التالي . كذلك من أخطاء المبرمجين الشائعة الدخول في دوار لا نهائي عند قراءة البيانات ويقوم البرنامج

حينئذ بعد انتهاء أسطر البيانات بمحاولة قراءة أسطر العمل التالي؛ لأن أمر القراءة مستمر بسبب خطأ الدوار اللانهائي، ولكن لأن أسطر العمل التالي تبدأ بعلامة ($). وهي علامة أوامر المراقب المقيم سيقوم برنامج المراقب المقيم بمعاملة هذه الحالة معاملة خطأ في البرنامج ويقوم تلقائياً بمسحه من الذاكرة والانتقال إلى العمل التالي في الحزمة . إذن ليست هناك مشكلة إذا عمل المبرمجون عبر برنامج المراقب المقيم، ولكن بعضهم لا يلتزم بذلك، ومن ثم لا يستطيع المراقب المقيم التحكم في أخطائهم مما ينتج عنه دخول الحاسوب في دوار مما يضطر لايقافه لمعالجة هذه المشكلة. ثم إضافة ثنائية في جهاز الحاسوب (الهاردوير) لتفرق بين أوامر المستخدم وأوامر المراقب حيث لا يقبل الحاسوب أسطر المراقب المقيم كمدخلات في برنامج المستخدم على الإطلاق. إذن تمكين الحاسوب منذ البداية من التفريق بين حالة المستخدم (user mode) وحالة المراقب أو المشرف (supervisor mode) يؤدي إلى عدم قبول أوامر برنامج المراقب المقيم كمدخلات في البرنامج، ومن ثم يتم تنفيذ تلك الأوامر التي تقوم بالتحكم في أخطاء المبرمجين وإلغاء برامجهم من الذاكرة والانتقال للأعمال التالية .

٥. نداءات النظام:

ما دام من غير المسموح للمبرمج التعامل مع أجهزة الإدخال والإخراج إلا عبر المشرف – فإن أجهزة الحاسوب الحديثة جعلت هناك أوامر تسمى نداءت المراقب أو المشرف، أو نداءات النظام يمكن لبرنامج المستخدم استدعاءها إذا رغب المستخدم في التعامل مع أي جهاز من أجهزة الإدخال أو الإخراج . عندما يصل النداء أو الأمر إلى البرنامج المراقب أو المشرف فإنه يقوم بتنفيذ النداء . بهذا يكون البرنامج المشرف هو الذي يقوم نيابة عن برنامج المستخدم عن طريق نداءات النظام بتنفيذ كل أوامر المستخدم المتعلقة بالتعامل مع أجهزة الإدخال

والإخراج وبهذا يكون البرنامج المراقب أو نظام التشغيل له التحكم التام في نظام الحاسوب فالمستخدم لا يستطيع مباشرة التعامل مع الحاسوب. لتنفيذ أي عمل يجب أن يتم ذلك عبر نظام التشغيل المشرف . فنظام التشغيل المشرف يتأكد أولاً من أن عمل المستخدم المراد تنفيذه مسموح به ومقبول من ذلك المستخدم ثم بعد ذلك يقوم بتنفيذ العمل نيابة عن المستخدم ثم يعيد للمستخدم التحكم مرة أخرى فيما يليه من أوامر . لابد أن نلاحظ أن هناك أوامر أخرى غير التحكم في الإدخال والإخراج لا يمكن أن يسمح للمستخدم بتنفيذها مثل أمر إيقاف الحاسوب (HALT) وأمر التحول من حالة المستخدم إلى حالة المشرف أو الدخول في منطقة الذاكرة الخاصة بالبرنامج المشرف (أو نظام التشغيل) (memory protection) .

٦. ساعة الحاسوب (Timer):

لقد تم تصميم ساعة الحاسوب لمنع برنامج المستخدم من الدخول في دوار لانهائي وامتداد سيطرة البرنامج المشرف . حيث تقوم ساعة الحاسوب عبر نظام التشغيل بإيقاف البرنامج إذا تجاوز فترة معينة مثلاً ١/٦٠ ثانية ثم بعد ذلك يقرر البرنامج المشرف إما إعطاء برنامج المستخدم زمناً إضافياً أو إعادته له برسالة (خطأ فادح) (fatal error) .

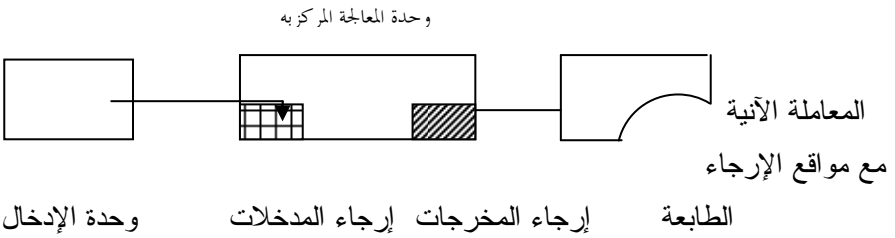
٧. الترجئة : (Buffering):

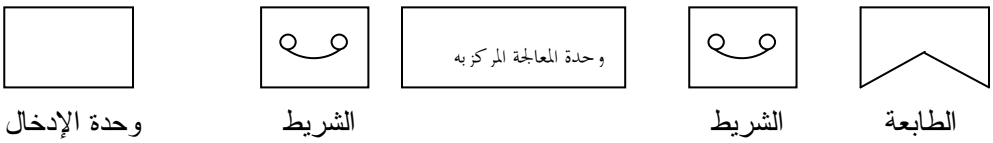
إن سرعة أجهزة الإدخال والإخراج لا تقارن بسرعة الحاسوب ولهذا ربما يظل الحاسوب عاطلاً بعض الوقت في انتظار المدخلات من أجهزة الإدخال وفي انتظار إخراج البيانات بواسطة أجهزة الإخراج . إن فكرة الترجئة تهدف إلى الاستفادة القصوى من وقت الحاسوب وأجهزة الإدخال والإخراج وذلك بحجز مواقع في الذاكرة للسجلات التي تتم قراءتها في انتظار المعالجة وتعرف هذه

المواقع بمواقع الإرجاء (buffers) وبهذا يمكن لجهاز الإدخال قراءة عدة سجلات ووضعها في مواقع إرجاء المدخلات قبل أن تدخل للمعالجة في وحدة المعالجة المركزية كما يمكن بعد إنهاء وحدة المعالجة المركزية من معالجة السجلات ثم إرسالها مباشرة إلى مواقع إرجاء المخرجات بدلاً من انتظار وحدة الإخراج لإخراج تلك السجلات . ولكن إذا كانت سرعة الحاسوب كبيرة جداً مقارنة بسرعة جهاز الإدخال فإن مواقع إرجاء المدخلات ستكون دائماً فارغة ومن ثم ليست ذات جدوى وكذلك إذا كانت سرعة الحاسوب كبيرة جداً مقارنة بسرعة جهاز الإخراج فإن مواقع إرجاء المخرجات ستكون ممتلئة وسيضطر الحاسوب للانتظار عاطلاً حتى تفرغ تلك المواقع . إذن فكرة الإرجاء لن تكون مفيدة إلا إذا كانت سرعة أجهزة الإدخال والإخراج مقاربة لسرعة الحاسوب في التعامل مع السجلات . ولما كانت سرعة الحاسوب عادة أكبر من سرعة أجهزة الإدخال والإخراج فإن إنهاء الأعمال يصبح مقيداً بسرعة أجهزة الإدخال والإخراج لا بسرعة الحاسوب ولكن في بعض الأحيان ربما تكون عمليات الحوسبة المطلوبة على السجلات عالية جداً وفي هذه الحالة ستكون مواقع إرجاء المدخلات ممتلئة ومواقع إرجاء المخرجات خالية بسبب الحوسبة العالية المطلوبة من وحدة المعالجة المركزية . بهذا يمكن القول إن فكرة الإرجاء أو الترجئة يمكن أن تعالج بعض المشكلة ولكن لن تعالج كل المشكلة بسبب الفوارق العالية بين سرعة الحاسوب وسرعة أجهزة الإدخال والإخراج وبسبب نوع الحوسبة المعقدة المطلوبة أحياناً .

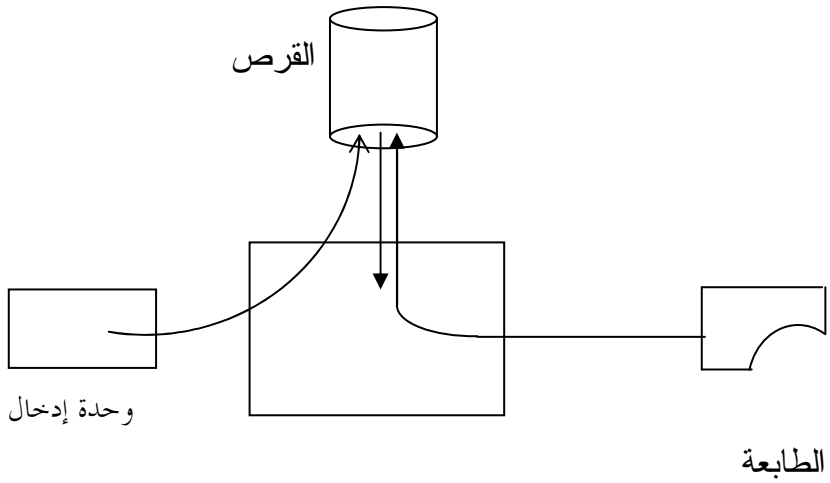
لمعالجة مشكلة الفوارق العالية بين سرعة الحاسوب وأجهزة الإدخال والإخراج يتم استخدام وسائط تخزينية أعلى سرعة مثل الشريط وتخزين البيانات عليه بعد إدخالها بواسطة جهاز الإدخال ثم يقوم الحاسوب بالتعامل مع السجلات في الشريط ومعالجتها ثم يقوم بإخراج البيانات علىالشريط قبل إخراجها بواسطة جهاز

الإخراج وهذه تعرف بالمعاملة غير الآنية . off-line operation (موضحة في الاشكال التالية)





المعاملة غير الآنية باستخدام الشريط



لقد تم تحسين فكرة المعاملة غير الآنية باستخدام الأقراص، والأقراص بالطبع لها ميزاتها على الأشرطة في أنها لا تشترط التعامل المتتالي. مثلاً لا يمكن على الشريط الكتابة والقراءة في وقت واحد فلا يمكن أن يتم إدخال البيانات عليه بواسطة وحدة الإدخال وفي نفس الوقت قراءة البيانات منه بواسطة الحاسوب او وحدة المعالجة المركزية لتقوم بمعالجتها، ولكن القرص يعالج هذه المشكلة حيث يمكن الكتابة عليه والقراءة منه آنياً حيث يقوم نظام التشغيل بوضع مدخلات كل عمل في جدول على التوالي الأول فالأول، ويقوم الحاسوب بقراءة تلك المدخلات حسب طلب البرنامج، ثم يقوم بنقل المخرجات إلى مساحة إرجاء المخرجات ليتم كتابتها على القرص لتتم طباعتها على التوالي الأول فالأول . هذه الطريقة تسمى المعاملة الآنية المشتركة (spooling) . في الواقع إن المعاملة الآنية المشتركة هي نفس فكرة الترجئة مع استخدام القرص بدلاً عن الذاكرة مما يمكن من تخزين الملفات بعد إدخالها في انتظار المعالجة أو تخزين الملفات بعد المعالجة في انتظار الإخراج وبهذا لن ينتظر الحاسوب عاطلاً لإدخال مدخلات أو عاطلاً لإخراج مخرجات .

وهنا يمكن إدخال مدخلات عمل وإخراج مخرجات عمل آخر أو معالجة عمل آخر في نفس الوقت. أما في حالة الترجئة فلا يتم إرجاء مدخلات أو معالجة برنامج أو إرجاء مخرجات إلا لنفس العمل إذ لا يمكن تجاوز عمل إلى عمل آخر في الإدخال أو المعالجة أو الإخراج . كذلك من المميزات الهامة للمعاملة الآنية المشتركة هو أنه بإمكان نظام التشغيل إعطاء أولوية لبعض الأعمال على الأخرى في الإدخال أو المعالجة أو الإخراج .

٨. البرمجة المشتركة : (Multi programming) :

تعمل البرمجة المشتركة على تقليل الوقت الضائع من الحاسوب ، فإذا كان الحاسوب يقوم بمعالجة برنامج عمل معين، وهناك بيانات طلبها البرنامج ولا زالت وحدة الإدخال تعمل على إدخال تلك البيانات فإن الحاسوب سيظل متعطلاً في انتظار تلك البيانات إذا كان بطريقة البرمجة المفردة، أما إذا كان يعمل بطريقة البرمجة المشتركة فإن هناك عدة برامج بالذاكرة في انتظار المعالجة بالحاسوب. ومن ثم لن ينتظر الحاسوب عاطلاً بل سينتقل فوراً إلى البرنامج التالي ويقوم بمعالجته أما ذلك البرنامج الذي لم تكتمل مدخلاته فسيعود إلى آخر الصف بالذاكرة وينتظر دوره ، فإذا جاء دوره وقد اكتملت بياناته ستتم معالجته وإذا لم تكتمل بياناته ستتم معالجته حتى لحظة الحاجة لمزيد من البيانات وإيقافه عند ذلك الحد ثم الانتقال للبرنامج التالي .

تعتبر البرمجة المشتركة من أهم ميزات نظم التشغيل الحديثة وقد تم تطويرها لإعطاء خدمة مريحة للمستخدم واستخدام أمثل لإمكانات الحاسوب ومن تلك التطورات الخدمة الدورية وهي أنه إذا تجاوز أي برنامج زمن معين فإن الحاسوب يوقف معالجته ويبدأ في معالجة البرنامج التالي ويعود ذلك البرنامج إلى آخر الصف . إن هذه الطريقة تضمن إنهاء الأعمال القصيرة فوراً وعدم تعطيلها بسبب الأعمال الطويلة . كما أنها تتجاوز تلقائياً أخطاء البرامج فإذا كان هناك برنامجاً به خطأ أدى إلى دوار غير نهائي فإنه لن يؤثر في الأعمال الأخرى أو يؤدي إلى تأخرها . إذ أنه بعد مضي بعض الوقت سيضطر الحاسوب إلى إيقاف ذلك العمل وعمل رسالة للمبرمج مفادها بأن هناك خطأ فادحاً في ذلك البرنامج .

٩. اشتراك وقت الحاسوب (الاستخدام المشترك):

لقد أتت فكرة حزمة الأعمال أو المعالجة الخدمية لمعالجة مشكلة إدخال وإخراج البرامج المساعدة والمترجمات التي يستخدمها المبرمجون، حيث يتم إدخال أعمال المبرمجين التي تستخدم نفس المترجمات أو البرامج المساعدة في شكل حزمة مع بعضها حتى يتم توفير وقت إنزال وإعادة المترجمات والبرامج المساعدة من الشريط مع كل برنامج على حده . ولكن بعد تطور تقنية الأقراص أصبح بالإمكان إنزال وإعادة أي مترجم أو برنامج مساعد يحتاج إليه المستخدم على الفور من والي القرص . فإنزال البيانات من القرص أو إعادتها إلي القرص لا يتم كما يحدث في الشريط بقراءة كل بيانات الشريط على التوالي حتى يتم الحصول على البيانات المطلوبة ثم تحويل الشريط من حالة القراءة إلى حالة التخزين ليتم إعادة تلك البيانات إلى الشريط وإنما يتم ذلك مباشرة بواسطة المؤشر الرأسي للقرص الذي يقفز مباشرة إلى موقع البيانات أو البرنامج المساعد أو المترجم المطلوب لتتم قراءته بواسطة الحاسوب كما يقفز مباشرة إلى المواقع الخالية ليتم تخزين البيانات المطلوب تخزينها .

بهذا لم تعد هناك حاجة للمعالجة الحزمية حيث أصبح كل مستخدم يتعامل مباشرة مع الحاسوب مستخدماً لوحة المفاتيح لإدخال البرامج ولإدخال البيانات والشاشة لاستقبال رسائل الحاسوب والمخرجات أو القرص لتخزين البرامج والبيانات واستدعاء البرامج المساعدة والمترجمات وغيرها كما يمكن للمستخدم عن طريق لوحة المفاتيح إعطاء الحاسوب أوامراً متاحة من نظام التشغيل تمكنه من التعامل مع أجهزة الإدخال والإخراج الأخرى كالماسحة والطابعة والراسمة والأنواع المختلفة من الأقراص والأشرطة. هذه الطريقة تعرف بنظام التخاطب المباشر (Interactive system) [لكن تظل هناك حاجة للمعالجة الحزمية في حالة

الأعمال الكبيرة التي تحتاج لزمن طويل لمعالجتها بواسطة الحاسوب ولا توجد حاجة للمستخدم على الشاشة لمراقبتها أو للتخاطب مع الحاسوب بشأنها فبإمكانه إرسال العمل في شكل حزمة ثم الذهاب لقضاء أعمال أخرى إلى حين إنهاء الحاسوب لذلك العمل ، لكن الواقع أنه لا توجد حزمة ، إنما هناك عمل واحد فقط يخص المستخدم كما أنه لا توجد ضرورة لوضع الأعمال في شكل حزمة مع وجود القرص كما أوضحنا سابقاً]. لهذا جاءت فكرة الجمع بين البرمجة المشتركة والتخاطب المباشر مع الحاسوب حيث يقوم كل مستخدم بإرسال عمله ويقوم الحاسوب بترتيب هذه الأعمال على التوالي الأول فالأول ولكنه لا يعطي اي عمل الزمن الكافي الذي ينهيه وإنما يعطيه زمن ثابت مثلاً ١/٦٠ ثانية فإذا لم ينتهى ذلك العمل يعود وينتظر في آخر الصف وهكذا .

بهذا يتم إنهاء الأعمال القصيرة فوراً ويتم تأخير الأعمال الكبيرة فالذين لهم أعمال صغيرة ينتظرون على الشاشة لمتابعة أعمالهم والذين لهم أعمال كبيرة يمكنهم بعد إرسالها الذهاب لقضاء بعض الأعمال والعودة مرة أخرى . هذه الفكرة في نظم التشغيل تعرف باشتراك وقت الحاسوب. وطريقة الخدمة التي يقدمها الحاسوب في نظام اشتراك وقت الحاسوب تعرف بالخدمة الدورية .

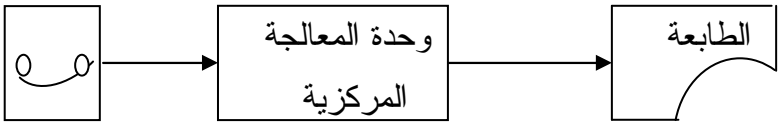
١٠. الأنظمة اللحظية : (Real-time systems)**:**

نعني بنظام البيانات اللحظية هو تحكم نظام التشغيل في وحدة إدخال بيانات خاصة بتطبيقات معينة مثل الحساسات المستخدمة في الأجهزة الطبية أو الحساسات المستخدمة في قراءة البيانات البيئية أو حساسات التصنت أو الحساسات المستخدمة في الصناعات أو المستخدمة في التجارب العلمية وغيرها . ثم يقوم الحاسوب بتحليل تلك البيانات ثم قد يوجه بضبط التحكم وطلب قراءة جديدة من الحساسة أو

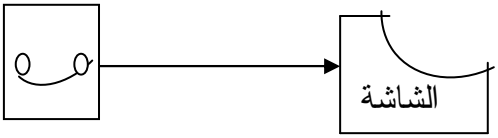
عمل المعالجة على تلك البيانات وإعطاء المخرجات المطلوبة على الفور . إذن أنظمة البيانات اللحظية تختلف من أنظمة اشتراك وقت الحاسوب والتعامل اللحظي مع البيانات وإذا لم يتم التعامل خلال وقت محدد سيؤدي ذلك إلى فشل النظام. وعليه فلا مجال لانتظار الأعمال في صف المعالجة كما هو الحال في اشتراك وقت الحاسوب .

١١. تطور أنظمة التشغيل

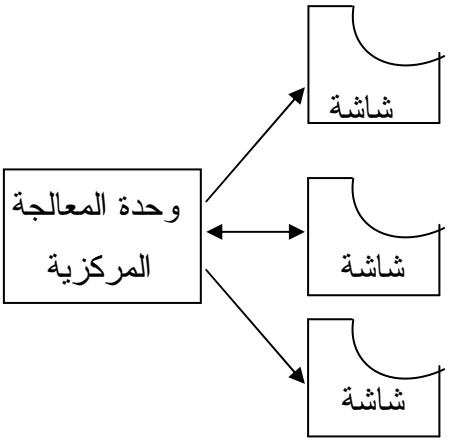
١) نظام المعالجة الحزمية



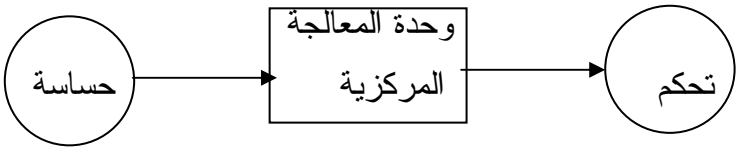
٢) نظام التخاطب المباشر مع الحاسوب



٣) نظام اشتراك الزمن



٤) النظام اللحظي



١٣. أنظمة المعالجات المشتركة : (Multiprocessing systems)**:**

بدلاً من استعمال وحدة معالجة مركزية واحدة جاءت فكرة أن تكون هناك وحدة معالجة مركزية رئيسة ومعالجات أخرى مساعدة لها تعني بأعمال محددة مثلاً أن تكون هناك وحدة معالجة مركزية مساعدة متفرغة للتعامل مع وحدات الإدخال والإخراج ووحدة معالجة مركزية أخرى متفرغة للتعامل مع شبكات الاتصال وتظل وحدة المعالجة المركزية الرئيسة متفرغة لمعالجة البرمجيات وتنفيذ الأعمال . هذه الفكرة تحفظ زمن المعالج الضائع في التنقل بين وحدات الإدخال والإخراج والاتصال والمعالجة .

الخلاصة :

نخلص من سرد هذا التطور التاريخي في بناء نظم التشغيل أن ذلك تم لتحقيق أهداف نظم التشغيل في الاستفادة القصوى من إمكانات الحاسوب وإعطاء الخدمة الممتازة للمستخدم سواء أكان ذلك في تسهيل عملية الاستخدام أم في إنهاء الخدمة في أقل وقت ممكن . لقد تم ذلك أولاً باستخدام فني تشغيل مدربين على إنزال وإخراج الأشرطة الممغنطة ليقوموا بهذا العمل في أقل وقت وليحفظوا وقت المبرمجين للبرمجة بدلاً من هذه الأعمال اليدوية. ثم تلا ذلك فكرة حزم الأعمال حيث يقوم المشغل بوضع الأعمال المتشابهة في شكل مجموعات ويتم تنفيذها

مجموعة تلو الأخرى ويقوم فني التشغيل بإدخال أعمال المبرمجين داخل المجموعة يدوياً . تلا ذلك أن يتم إدخال كل أعمال المبرمجين في المجموعة الواحدة مرة واحدة، وهناك أوامر تقوم بالفصل بين عمل وآخر وتحدد المترجمات المطلوبة لأي عمل، ومن ثم يتم تنفيذ كل الأعمال داخل المجموعة تلقائياً دون الحاجة لفني التشغيل وهذه كانت بداية فكرة نظام التشغيل. وقد سمي البرنامج المحتوي لهذه الأوامر بالمراقب المقيم . ثم طورت هذه الفكرة في صناعة الحواسيب حيث صممت ثنائية تصل بين أوامر المبرمج أو المستخدم وأوامر المشرف أو المراقب وذلك لتجاوز أخطاء المبرمجين وتدخلهم في أعمال المشرف أو ضبطهم في التعامل مع أجهزة الإدخال والإخراج وتجاوز صلاحياتهم في التعامل مع الذاكرة ووحدة المعالجة المركزية. بعد ذلك تمت إضافة ساعة الحاسوب عند صناعة الحاسوب لمنع خطإ الدوار اللانهائي الذي يحدث عادة مع المبرمجين . ثم تم إضافة حالتي المستخدم والمشرف وتحديد الصلاحيات وحماية الذاكرة. وبتصميم الساعة بدأ تصميم نظام التشغيل بمفهومه الحديث . فأدخلت فواصل الترجئة لتقليل الزمن الضائع من وحدة المعالجة المركزية. ثم أدخلت فكرة المعاملة غير الآنية بإدخال الأشرطة كوسائط بين وحدات الإدخال والإخراج وبين المعالجة المركزية، وأخيراً بعد تطور تقنية الإشراف ثم إدخال فكرة المعاملة الآنية المشتركة لما في الإشراف من ميزة التعامل المباشر . لقد تم تطوير فكرة المعاملة الآنية المشتركة إلى البرمجة المشتركة حيث أصبح الحاسوب يحتفظ بعدة برامج في الذاكرة في وقت واحد ويعالجها على التوالي. وأخيراً أدت فكرة البرمجة المشتركة إلى الاستخدام المشترك الذي يمكن مئات المستخدمين من التخاطب مع الحاسوب ومعالجة برامجهم في وقت واحد .

١٣. نظام التشغيل يونيكس

لأن يونيكس يمثل الأساس لكل أنظمة التشغيل الحديثة فلابد من إعطاء بعض الأوامر الهامة المستخدمة في هذا النظام. في البداية يجب ان يدخل اسمه وكلمة سره الابتدائية للمشرف على النظام لأنه عندما يريد استخدام النظام سيواجه بالأمر:

- ادخل اسمك المعرف لدى المشرف **Login : Ali**

بعد الضغط على مفتاح "ادخل" سوف يظهر لك الأمر :

- ادخل كلمة السر **Password : SUDAN1**

إذا أخطأت في الاسم ستظهر لك رسالة توضح بأن هذا المستخدم غير معروف، وعليك إعادة كتابة الاسم أما إذا أخطأت في كلمة السر فسيظهر لك خطأ ويطلب منك إدخال كلمة السر مرة أخرى . وإذا كان الاسم صحيحاً وكلمة السر صحيحة سوف تظهر لك رسالة ترحيب ويكون الحاسوب في انتظار أوامرك .واذا كتبت أي امر غير معروف مثل "**eat fool**" سيرد عليك الحاسوب :

هذا الأمر غير موجود	**eat: command not found**

فيما يلي سنسرد بعض الأوامر الهامة في يونيكس :

- **الأمر pwd :**

يظهر الدليل الحالي ويستفاد منه في تحديد موقعك قبل قيامك بأي عمل في الدليل الحالي

$ pwd
$ /home/ali

هذا يعني أن الدليل الحالي هو /home/ali

- **الأمر passwd :**

لتغير كلمة السر (يوصي بأن يغبر المستخدم كلمة السر كل فترة حتى لا تتسرب) ويتم ذلك بالأمر .

```
$ passwd
Changing password for ali
Old password :
New password :
```

سيرد لك الحاسوب إدخال كلمة السر القديمة حتى يتأكد أنك نفس الشخص المخول له تغيير كلمة السر بعد إدخالك الكلمة القديمة والتاكد منها يطلب منك الحاسوب إدخال كلمة السر الجديدة وتصبح هذه هي كلمة السر التي تمكنك من دخول الحاسوب .

- **الأمر mkdir :**

لإنشاء الأدلة الفرعية في الدليل الحالي بعد أن أمنت نفسك يمكنك الآن تنظيم بياناتك ويتم ذلك بجعل كل الملفات التي لها علاقة مع بعضها البعض تكون في دليل واحد خاص بها . مثلاً ملفات قسم الحاسوب تكون في دليل اسمه الحاسوب "computer" يتم تكوين ذلك الدليل على النحو التالي :

```
$mkdir computer
```

- **الأمر cd :**

للتنقل بين أدلة النظام ، فإذا أردنا الانتقال من الدليل الحالي وهو "ali " إلى الدليل الحاسوب "computer" فإن ذلك يتم بالأمر:

```
$cd computer
$ pwd
$/home/ali /computer
```

أي أن أوامرنا أصبحت الآن في الدليل الفرعي"computer" وإذا أردنا العودة مرة أخرى إلى الدليل "ali " فإننا نكتب الأمر :

```
$cd ..
$ pwd
$/home/ali
```

إذا عملنا دليلاً آخر للرياضيات يسمي "math "

```
$mkdir math
```

- **الأمر ls :**

لمعرفة كل الملفات والأدلة الموجودة في الدليل الحالي وهو "ali " فإن ذلك يتم بالأمر :

```
$ ls
.
..
computer
math
$
```

- **الأمر cp :**

يستخدم لنسخ ملف في ملف آخر أو إلى دليل آخر . إذا كان هنالك ملف في الدليل الحالي يسمي "file1 " ونريد نسخه في الملف "file2 " فيتم ذلك بالأمر التالي

```
$cp file1  file2
```

أما إذا أردنا نسخه في الدليل الفرعي "computer" فيتم كالأتي:

```
$cp file1  computer
```

أما إذا كان الملف غير موجود فإنك ستحصل علي رسالة خطأ بالشكل

```
$cp file3  file2
cp: cannot access file3
```

- **الأمر mv :**

يقوم هذا الأمر بنقل أو إعادة تسمية الملفات والأدلة .

معلومة

عملية النقل تختلف عن عملية النسخ ،لأن النسخ يبقي الملف الأصلي كما هو وينشي ملفاً آخراً مماثل له بينما النقل ينشئ ملفاً آخراً ويلغي الملف الأصلي

```
$ ls
file1
file2
$ mv file1 file4
$ ls
file4
file2
```

- **الأمر rm :**

يقوم هذا الأمر بحذف الملفات

```
$ ls
file4
file2
$ rm file4
$ ls
file2
```

- **الأمر rmdir :**

يقوم هذا الأمر بحذف الأدلة بشرط أن تكون هذه الأدلة فارغة

```
$ rmdir computer
rmdir :computer not empty
$ cd computer
$ rm file1
$ cd ..
$ rmdir computer
```

- **الأمر more :**

يقوم هذا الأمر بعرض ملف نصي علي الشاشه بشكل صفحات متتالية يتم التوقف فيما بينها وهذا الامر يشبه الأمر **cat** مع الاختلاف في التوقف بين الصفحات .

```
$ more file2
```

- **الأمر clear:**

يقوم هذا الأمر بمسح الشاشة وإظهارها خالية مع وضع المؤشر وإشارة النظام في الزاوية العليا اليسارية للشاشة.

- **الأمر exit:**

يستخدم هذا الأمر لإنهاء عمل إجراء ما والخروج من مستوي الي مستوي أعلي منه .

- **الأمر factor:**

يقوم هذا الأمر بإيجاد العوامل الأولية للأعداد الصحيحة الموجبة

```
$ factor
72
    2
    2
    3
    3
    3

$ factor
150
    2
    3
    5
    5
```

- **الأمر rev:**

يقوم هذا الأمر بعكس ترتيب الحروف في كل سطر من ملف ما فمثلاً اذا كان الملف"file2 " يحتوي علي العبارة التالية :

"programming in logic"

يقوم هذا الأمر بعكس العبارة لتصبح

"cigol ni gnimmargorp"

```
$ cat file2
programming in logic
$ rev file2
cigol ni gnimmargorp
```

- **الأمر cal:**

يستخدم هذا الإمر لإظهار التقويم الميلادي

```
$ cal 3 2002
March 2002
 S   M   Tu  W   Th  F   S
                     1   2
 3   4   5   6   7   8   9
 10  11  12  13  14  15  16
 17  18  19  20  21  22  23
 24  25  26  27  28  29  30
 31
$
```

عند استخدمه من غير تحديد الشهر والعام سوف يقوم بإظهار التقويم الميلادي للشهر الحالي فقط .

- **الأمر grep:**

إذا أردت البحث عن سلسلة من الحروف في ملف او أكثر من ملف سوف يتم ذلك بهذا الأمر وستكون عملية البحث سطراً سطراً. فمثلاً اذا كان :

الملف "file1" يحتوي علي المعلومات التالية

```
mohammed osman
shemma hassan
rowa abd alla
```

الملف "file2" يحتوي علي المعلومات التالية

```
mohammed osman
hadeal mamoon
```

```
$ cat file1
mohammed osman
shemma hassan
rowa abd alla
$ cat file2
mohammed osman
yousif ali
$ grep "mohammed osman" file1
mohammed osman
$ grep "mohammed osman" file1 file2
file1:mohammed osman
file2:mohammed osman
```

- **الأمر man :**

إذا أردت المساعدة من يونيكس بتذكيرك بالأوامر أو طريقة استخدامها فإن هذ الأمر يمكنك من ذلك فمثلاً :

```
$ man ls
```

يتم عرض كل المعلومات الخاصة باستخدام الأمر **ls** ويمكن تكرار هذا الأمر مع جميع الأوامر الأخرى للتأكد منها .

١٤. نظام التشغيل لينوكس :

بعد تطور أنظمة الحواسيب الشخصية والتي بدأت في استخدام المعالجات ذات الـــ ٣٢ ثنائية لم يعد نظام التشغيل دوس مناسباً لتشغيلها التشغيل الأمثل، فتم استبداله بنظام التشغيل يونيكس و نظام التشغيل وندوز إن تي الشبيه بيونيكس . ولكن أسعار يونيكس ووندوز إن تي غير مجدية لكثير من مبرمجي ومستخدمي الحواسيب الشخصية من ثم بدأ هؤلاء المبرمجون والمستخدمون يبحثون عن بدائل أخرى كان أنسبها نظام التشغيل لينوكس .

لقد تم تصميم نظام التشغيل ليونيكس بواسطة طالب فنلندي يدعى لينوس تورفالدس عام ١٩٩١م بخواص يونيكس ثم طرحه مجاناً لعامة الناس على الانترنت وطلب من كل المبرمجين ومصممي نظم التشغيل المشاركة في تطوير هذا النظام مما أدى إلى تطويره سريعاً . إذن نظام لينوكس هو نظام تشغيل غير تجاري متاح مجاناً للاستخدام ويشارك الجميع في تطويره من خلال مجموعات عمل عبر الانترنت . لقد تم طرح النسخة العملية الأولى للينوكس في مارس ١٩٩٤ ثم النسخة الثانية في يونيو ١٩٩٦ .

يتميز لينوكس:

- بإمكانية التعامل مع كل البرمجيات التي تم تصميمها على أنظمة التشغيل الأخرى مثل يونيكس ووندوز مايكروسوفت ودوس ومع لغات البرمجة المختلفة .

- كذلك التعامل مع نظام أكسس وندوز بخواصه المتطورة في التخاطب المباشر مع ربط الصور . كما له خاصية توسيع الذاكرة .

- لقد استفاد لينوكس من النسخة القياسية لنظام التشغيل يونيكس مما مكنه من التعامل مع بيئات يونيكس المختلفة والاحتفاظ بخاصية يونيكس في الاستخدام المشترك والبرمجة المشتركة (multiprogramming) .

- هذا إضافة إلى إمكانية التعامل مع تقنيات وبرتوكلات شبكات الاتصال المختلفة مثل تي سي بي / آي بي (TCP/IP) .

أولاً :استخدامات لينوكس :

لقد تم تصميم لينوكس كما ذكرنا على معمارية الـــ ٣٢ ثنائية بمرونة لا تقيده على معمارية بعينها فهو الآن يمكن استخدامه على كل المعماريات المتاحة والمستقبلية بسهولة ويسر كما أن له القدرة في التعامل مع المعالجات المركبة (multiprocessing) والتعامل اللحظي (real time) . إذن يمكن القول أن نظام لينوكس يمكن استخدامه في كل التقنيات المتاحة والمستقبلية لأجهزة الحواسيب بإمكانات عالية تفوق كل أنظمة التشغيل الحالية . يمكن استخدام لينوكس مع المخدمات ومع الطرفيات ومع أجهزة الذكاء الاصطناعي (الروبوتات) والحساسات وأجهزة التحكم ومع أجهزة الحواسيب الكبرى . هذا إضافة إلى استخدامات لينوكس المتطورة على برمجيات خدمات الانترنت وفي قدرته على حماية البيانات بواسطة ما يعرف بالحوائط النارية (fire walls) .

إن للينيوكس المرونة الكافية في التعامل مع حاسوب ضعيف الامكانات مثل معالج انتل ٣٨٦ وبذاكرة ٤ م ب فقط وإذا أردنا التعامل مع الرسوم التي تحتاج لامكانات ذاكرة أكبر فإننا نحتاج كحد أدنى إلى ٨ م ب وتزيد الحاجة إلى ذاكرة أكبر ومعالج أقوى إذا أردنا تعاملاً أدق مع الرسوم وقد يطلب لينوكس حسب

متطلبات الرسومات والبرمجة ٦٤ م ب أو ١٢٨ م ب هذا بالطبع مع معالجات بنتيوم ٣ كحد أدنى في هذه الحالات .

ثانياً : تشغيل لينوكس :

يمكن الحصول على لينوكس مجاناً على أقراص مرنة أو أقراص مضغوطة وحجمه حوالي ٤٠ م ب في نسخته الصغرى (أي ليس بها إمكانات الرسومات) وحوالي ١٢٠ م ب في النسخة الكبرى . لذا يتضح أنه ليس عملياً نسخه في أقراص مرنة؛ لأن ذلك يتطلب كحد أدنى حوالي ١٠٠ قرص مرن ، لذا من العملي والأجدى أن يتم اقتناء لينوكس على قرص مضغوط .

عند إنزال لينوكس من القرص إلى الجهاز هناك أسئلة سوف يطلب منك لينوكس الإجابة عليها تخص التقنية المستخدمة في ذلك الجهاز . أولاً سوف يسألك عن متحكمات الأقراص الصلبة المستخدمة وهي عادة المتحكم الأول به قرص أساسي يسمى C وقرص تابع يسمى D والمتحكم الثاني وبه قرص رئيس يسمى E وقرص تابع يسمى F . وهذه المتحكمات هي في الغالب من نوع IDE .

هناك متحكمات من نوع آخر يعرف بـــ SCSI لكنها غير منتشرة مع الحواسيب الشخصية . بعد ذلك سوف يسأل لينوكس هل تريد وضع نظام التشغيل في قرص منفصل أم مع الأنظمه الأخرى والأفضل أن يكون في قرص منفصل وفي القرص الأول C . ربما يسأل كذلك لينوكس أثناء الإنزال عن عدد المسارات في القرص وعدد الاسطح وعدد المقاطع في المسار الواحد وهذه المعلومات تكون عادة مطبوعة على القرص من الخارج .

ثالثاً : بعض أوامر لينوكس الهامة وما يقابلها في UNIX وDOS

DOS	UNIX	LUNIX
dir	ls	Ls / dir
cls	clear	clear
del	rm	rm
copy	cp	cp
move / rename	mv	mv
type	cat	cat
cd	cd	cd
more < file	more file	more file
md	mkdir	mkdir
rd	rmdir	Rmdir

١٥. تمرين

١) عرف نظام التشغيل ثم صف بالرسوم نظام التشغيل كنظام فرعى للحاسوب .

٢) كيف كان تنفيذ البرامج فى بداية اختراع الحاسوب؟

٣) ما التطور والتحسين الذى حدث فى تنفيذ البرامج بعد تصميم المترجمات؟

٤) ما الظروف التى أدت الى إدخال وظيفة المشغل؟ وما دوره؟ وما التحسين الذى طرأ عليه بعد ذلك ؟

٥) ما فكرة حزم الأعمال وما الدافع من ورائها ؟

٦) صف التوالى التلقائى، وما الهدف من ورائها؟

٧) ما اسم البرنامج الذى يعتبر أول خطوة فى تصميم برنامج التشغيل؟ أعط مثالاً لأوامر ذلك البرنامج، وكيف يفرق بينها وبين أوامر برنامج المستخدم .

٨) اعط أمثله للحالات التى يقع فيها برنامج المراقب المقيم فى مصيدة.

٩) ماذا يحدث مالم يلتزم المبرمج بأوامر برنامج المراقب المقيم؟ كيف تم معالجة هذه المشكلة؟

١٠) صف فكرة نداءات النظام ومبرراتها .

١١) ما الأوامر الأخرى التى لايسمح للمستخدم باستخدامها؟

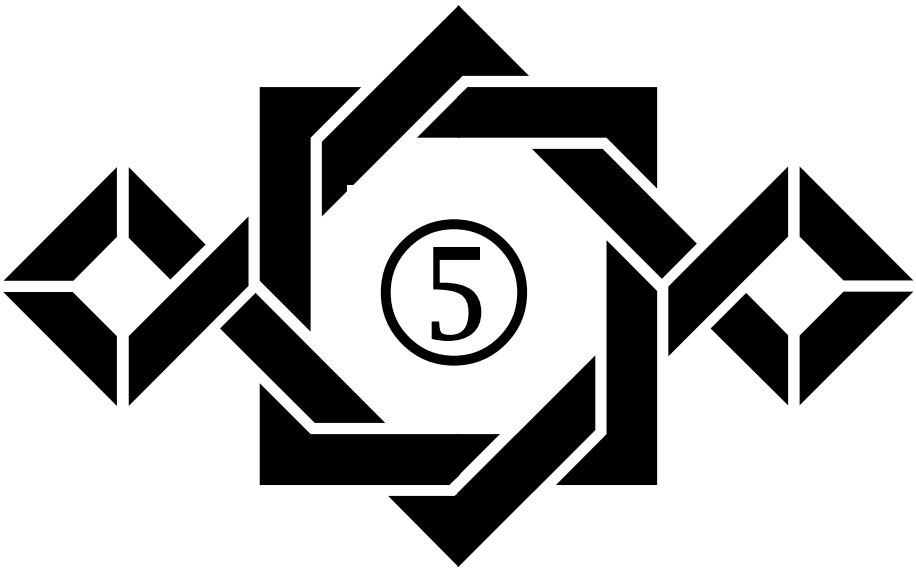
١٢) لماذا تم تصميم ساعة الحاسوب؟

١٣) ما ميزة فكرة الترجئة ؟ وما نقاط ضعفها؟

١٤) كيف تمت معالجة مشكلة ضعف فكرة الترجئة وبماذا تعرف هذه المعالجة؟

١٥) صف فكرة المعاملة الآنيه، وما أهمية القرص فى هذه المعاملة؟ وماالفرق بينهما وبين فكرة الترجئة؟

١٦) صف ميزات المعاملة الآنية.
١٧) صف نظام البرمجة المشتركة وميزاتها.
١٨) صف نظام الاستخدام المشترك، وما اهم تقنية تخزين مستخدمه فيه؟
١٩) صف نظام التخاطب المباشر.
٢٠) ما الحاجة للمعالجة الحزمية بعد دخول نظام التخاطب المباشر؟
٢١) ما الهدف وراء فكرة الجمع بين نظامى البرمجة المشتركه والتخاطب المباشر؟
٢٢) عرف الانظمة اللحظية واعط أمثلة لاستخداماتها.
٢٣) ما الفائدة من أنظمة المعالجة المشتركة؟
٢٤) بين بالشرح المختصر وبالرسم تطور أنظمة التشغيل .
٢٥) ما مبررات تصميم نظام لينوكس.؟
٢٦) كيف تم تصميم نظام لينوكس كنظام غير تجارى؟ وكيف تتم عملية تطويره ؟
٢٧) أذكر خواص لينوكس الاربع الهامة .
٢٨) أذكر خمس من أوامر يونيكس الهامه واشرحها .
٢٩) قارن تلك الأوامر مع أوامر لينوكس ومع أوامر دوس.
٣٠) ما أهم ميزة فى استخدام يونيكس؟
٣١) ما الغرض من الأمر **Factor** في يونيكس وهل له مقابل في نظام دوس.
٣٢) ما اهم الاسئلة عند تشغيل يونيكس ؟
٣٣) اذكر ثلاث أوامر فى إدخال البيانات وفى التعامل مع الأدله وفى التعامل مع الملفات فى نظام يونيكس .



الذّكاء الاصطناعيّ ولغة برولوق

الذّكاء الاصطناعي ولغة برولوق

١. مدخل :

علم الذكاء الاصطناعي هو أحد علوم الحاسوب الذي يعني بالمعالجات الرياضية والمنطقية التي تجعل من الحاسوب آلة ذكية. ويعنى بالذكاء القدرة على استيعاب الحقائق وعلاقاتها مع بعضها واستخدام تلك الحقائق والعلاقات للتوصل إلى نتائج صحيحة ومنطقية لأقصى حدٍ ممكن ، فعلى سبيل المثال إذا غذينا الحاسوب بصورة أي طالب عن طريق الكاميرا الرقمية ومع صورة الطالب نداء صوتي لاسم الطالب فإننا نستطيع أن نجعل الحاسوب ينادي اسم الطالب كلما وقف الطالب أمامه ويتم ذلك على النحو التالي : عند وقوف الطالب أمام الحاسوب يقوم الحاسوب بأخذ صورة رقمية لذلك الطالب ثم مضاهاة تلك الصورة مع الصورة التي تم تغذيته بها وينادي الحاسوب الطالب متى ما تطابقت تلك الصورة مع صورة الطالب والتي سبق أن تم تخزين نداء اسم الطالب معها .

مثال آخر يمكن أن يغذى الحاسوب بقوانين لغة الشطرنج والتحرك الصحيح مقابل أي حركة من اللاعب المنافس، وبهذا لا يمكن أن ينهزم الحاسوب أمام اللاعب المنافس وإذا كان المنافس ماهراً فإن اللعبة لن تنتهي أبداً . ومثال آخر يمكن أن ينصح الحاسوب بعدم السفر بالسيارة بناء على بيانات مناخية يتم تغذيته بها وتمكنه من تحليل المناخ والتنبؤ بحدوث أمطار غزيرة أو رياحاً كثيفة . مثال آخر هو أن يتم تصنيع آلة تشبه الإنسان بها كاميرا رقمية وحاسوب يكون الحاسوب بمثابة المخ والكاميرا بمثابة العين فمتى ما صورت الكاميرا جسماً أمامها يوجه الحاسوب تلك الآلة بالتوجه يميناً أو شمالاً أو للخلف كما يمكن تعقيد هذه الآلة بحيث إن رأت صورة أحد الطلاب الذين سبق أن غذيت بصورهم فإنها تنادي

الطالب باسمه أن افسح الطريق، كما يمكن لهذه الآلة بأن تستخدم في العمليات الخطيرة مثل القصف الجوي أو معالجة التسرب النووي أو إطفاء الحرائق أو في التجارب المؤثرة على الصحة مثل التجارب النووية ذات الإشعاع العالي أو التجارب التي تتم في بيئة يستحيل على الإنسان التعايش معها .

٣. أهداف علم الذكاء الاصطناعي وأمثلة لتطبيقاته :

يهدف علم الذكاء الاصطناعي إلى :

- تصميم وبرمجة آلات ذكية تقوم بتنفيذ الأعمال المطلوبة منها بكل دقة مثل قيادة الطائرات ووصول الصواريخ للأهداف المقصودة .
- برمجة الحقائق وعلاقاتها بصورة تؤدي إلى نتائج منطقية وإلى تحقيق أهداف صحيحة مثل تحليل وتخزين كلمات وتراكيب اللغات لأجل الترجمة الفورية وتحليل وتخزين بيانات المجرمين لمعرفة المتهمين في جريمة معينة أو في التخطيط الدقيق بناء على تحليل لحجم ضخم من المعرفة والمعلومات.
- يمكن للحاسوب ان يلعب دور الخبير البشري في الفتوي في مجال الخبره فيقوم بتشخيص الاعطاب او الأمراض أو الإرشاد والنصح او التنبؤ أو التخطيط . . . إلخ.
- استخدام النماذج المحوسبة لدراسة سلوكيات الإنسان أو الحيوان أو الأحياء عموماً .
- استخدام الحاسوب مثله مثل الإنسان الواعي المتعاون – مثل معرفة الداخلين في الأماكن الحساسة من خلال تحليل الصورة أو الصوت والسماح للمأذون لهم بالدخول أو استخدام الإنسان الآلي في إطفاء الحرائق او أعمال يدوية دقيقة (روبوت) .

- استخدام الحاسوب في مساعدة الإنسان المعاق باستخدامه في التحكم في يد صناعية أو رجل صناعية أو مساعدة قلب ضعيف الحركة .

٣.أسس علم الذكاء الاصطناعي :

لتحقيق أهداف علم الذكاء الاصطناعي التي ذكرناها في الفقرة السابقة هناك مجموعة من المعارف الضرورية التي يجب أن يلم بها مهندس برمجيات الذكاء الاصطناعي أول تلك المعارف بالطبع هي نظم الحاسوب (تقنية أجهزة وتقنية برمجيات) وعلم المنطق والنظريات الرياضية في الحوسبة وعلم النفس أو تركيب النفس البشرية ونظريات التحكم الالكتروني والنظريات الرياضية في حل المشكلات إضافة إلى المعرفة بالاقتصاد . في الغالب لا نجد شخصاً يلم بكل تلك المعارف لذا كان تصميم وبرمجة أي نظام يقوم على الذكاء الاصطناعي هو عمل مجموعة خبراء وليس عمل فرد حتى نضمن نجاح التصميم الذي يتمكن فعلاً من التحكم في حركة الإنسان الآلي (روبرت) بناءً على خطة دقيقة وقدرة على الإحساس بالنظر أو اللمس أو بالشم أو غيرها من المؤثرات الالكترونية تمكنه من :

- تفسير وفهم اللغات الإنسانية المكتوبة أو المسموعة .
- تفسير المعلومات بالوصول إلى نتائج هامة سواء أكانت طبيعية أم بيئية أم أمنية أم نفسية .

ويتم ذلك بأقل تكلفة ولتحقيق أكبر فائدة اقتصادية ممكنة .

٤. لغات الذكاء الاصطناعي :

إن المدخل إلى الذكاء الاصطناعي لا يتم إلا بنوع خاص من لغات الحاسوب العليا تعرف بلغات برمجة المنطق التي تساعد في جعل الحاسوب يتعامل بطريقة منطقية أو كأنه آلة مدركة ، وبهذا تكون لغات برمجة المنطق هي اللغات المناسبة في تطبيقات الذكاء الاصطناعي . إن كانت اللغات العالية الآن تطورت لتلائم هندسة البرمجيات بالتوجه نحو الشيئيات مثل C++ وغيرها فإن لغات برمجة المنطق أصلاً صممت لتلائم هندسة البرمجيات بالتوجه نحو المطلوب أو نحو الأهداف وهذا هو النوع من البرمجيات الذي يحقق أغراض الذكاء الاصطناعي إذ أن غاية الذكاء الاصطناعي أن يدرك الحاسوب ما المطلوب منه أو ما الأهداف التي يجب أن يحققها بناء على معلومات معينة أدخلت عليه ؟

إن أهم لغات الذكاء الاصطناعي لغتان لغة لسب **(Lisp)** ولغة برولوق **(Prolog)** . لقد تم تصميم لغة لسب عام ١٩٨٤م بمعهد ماساشيوتس المشهور بـــ **(M. I. T.)** بمدينة بوسطن بالولايات المتحدة الأمريكية خصيصاً لتحقيق الأغراض البرمجية للذكاء الاصطناعي . أما لغة برولوق فقد تم تصميمها في عام ١٩٧٠م بجامعة مرسيليا بفرنسا بغرض برمجة المسائل المنطقية قبل ظهور علم الذكاء الاصطناعي بصورة واضحة، ومن ثم أصبحت لها قدرة عالية في برمجة تطبيقات الذكاء الاصطناعي تفوق تلك التي في لسب إلا أن لسب ظلت الأكثر استخداماً لتطورها في أمريكا أكثر بلدان العالم استخداماً للحاسوب . إننا لسنا في حاجة لمقارنة اللغتين ولكن لغرض هذا المنهج سوف نكتفي بوصف مبادئ لغة برولوق وطريقة استخدامها مع أمثلة بسيطة من أمثلة الذكاء الاصطناعي .

٥. لغة برولوق :

لقد تم تصميم لغة برولوق (**PROLOG**) في جامعة مارسيليا بفرنسا في عام ١٩٧٠ بواسطة عالم الحاسوب الفرنسي آلان كولمراور.وكلمة برولوق مشتقة من الحروف الثلاث الأولي لكلمة برمجة (**PRO**gramming) والحروف الثلاث الأولي لكلمة منطق (**LOG**ic). إذن كلمة برولوق هي إختصار لكلمتي البرمجة بالمنطق أو برمجة المنطق وهذه الخاصية للغة برولوق – أي برمجة المنطق – تمكن من تحويل الحاسوب لآلة ذكية بمعني أن له القدرة في تخرين مجموعة حقائق ثم إستخدام تلك الحقائق في الوصول إلي نتائج وقرارات ذكية. مثلاً أن يتم تخرين كل المعلومات المتاحة عن الخصائص السلوكية والخلفيات البيئية والإجتماعية والأمزجة الشخصية لكل الطلاب، ثم بناء على هذه المعلومات يقوم الحاسب بتقسيم الطلاب إلى مجموعات صغيرة متجانسة لغرض السكن في غرف مشتركة، أو داخليات مشتركة. مثال آخر بأن يتم تخزين معلومات شرطية عن كل النشالين في الولاية وتشمل المعلومات طريقة نشلهم، وأماكن وجودهم، وأعمارهم، وألوانهم، وأشكالهم، فإذا حدثت شكوى نشل من مواطن يتم بناءً على بيانات المواطن تحديد أقرب النشالين تطابقاً مع بيانات الشاكي، ثم تكمل الشرطة تحرياتها لتحديد النشال . ونفس الطريقة يمكن أن تستخدم في تحديد سارقي المنازل، وسارقي المتاجر، والقتلة وغيرهم.

العلم الذي يساعد في تحويل الحاسوب إلى آلة ذكية يعرف في علوم الحاسوب بالذكاء الاصطناعي (Artifitial Intelegence) وتسمى اللغات التي تبرمج الحاسوب ليكون آلة ذكية بلغات الذكاء الإصطناعي وأهم لغة في الذكاء الإصطناعي اليوم هي لغة برولوق .

تعتبر لغات الذكاء الإصطناعي عموماً أكثر كفاءة من اللغات التقليدية مثل بيزك وباسكال ونعني بالكفاءة تقليل زمن تنفيذ البرنامج وتقليل حجم التخزين في

الذاكرة. لكن في المقابل يحتاج برنامج لغة الذكاء الإصطناعي إلى مجهود ذهني أكبر من المبرمج في تحديد كل الحقائق والبيانات الواقعية (Facts) ثم تنظيم تلك الحقائق وربطها ببعضها (Rules) وتوجيهها لاستخلاص النتائج والأهداف المطلوبة (goals). أما اللغات التقليدية فتقوم بتوجيه الحاسب إلى تنفيذ المطلوب في شكل خطوات تسلسلية أو دوائر تكرارية كما شرحنا ذلك في كتاب السنة الثانية عند وصف طرق البرمجة.

٦. بنائية لغة برولوق :

تعتبر لغات الذكاء الاصطناعي عموماً أكثر بنائية من اللغات التقليدية لعدم إستخدامها الدوائر الإلكترونية وهذا يجعلها أكثر وضوحاً وأقل عرضة للخطأ لكنها في المقابل كما ذكرنا تحتاج لحجم كبير من الحقائق والعلاقات التي لا بد أن يغذى بها الحاسوب قبل البدء في التنفيذ أو المعالجة.

تتكون بنائية لغة برولوق من الأجزاء التالية :

أولاً : تعريف أسماء وأشكال وأوصاف الأشياء المتعلقة أو ذات الصلة بالموضوع

(مثال ١) :

إذا كان الموضوع هو المدرسة فإن الأشياء التي يمكن ان تكون ذات العلاقة بالمدرسة ؟

هي :

- اسم المدرسة .
- المحلية التي تتبع لها .
- المرحلة التعليمية .
- عدد الفصول .
- عدد الطلاب .
- عدد الأساتذة .

ثانياً: تعريف ووصف العلاقات التي تربط بين تلك الأشياء التي تم تعريفها في المثال السابق يتم تعريف العلاقة التالية :

المدرسة(اسم المدرسة،المحلية،المرحلة،عدد الفصول،عدد الطلاب،عدد الأساتذة).

ثالثاً: وصف الحقائق والقوانين التي أسست عليها هذه العلاقات، مثلاً من الحقائق يمكن أن تكون:

المدرسة (أبو حشيش، البر الشرقي،أساس ، ٨ ، ٣٠٠، ١٠).

أي أن :

- اسم المدرسة هي أبو حشيش .
- اسم المحلية هي البر الشرقي .
- عـــدد الفصول= ٨ .
- عدد الطلاب =٣٠٠ .
- عدد الأساتذة= ١٠.

رابعاً: إستخدام هذه الحقائق والقوانين في الوصول إلى الأهداف المطلوبة، مثلاً أن نسأل عن عدد الأساتذة أو عدد الطلبة أو الفصول في مدرسة أبو حشيش أو أي مدرسة أخري تم وصفها .

إذا قارنا لغة برولوق بلغة باسكال نلاحظ أن الجزء الأول والجزء الثاني في لغة برولوق هما جزئين تعريفيين يقابلان جزء إعلان الأسماء والمتغيرات والدوال والآجراءات في لغة باسكال أما الجزء الثالث والجزء الرابع واللذان يقابلان الجسم الرئيس في لغة باسكال فيختلفان إختلافاً جذرياً في البنائية المنطقية كما في المثال التالي:

(مثال ٢) :

ادرس العبارات التالية جيداً

الحقائق : يحب محمد الفول

يحب علي العدس

القانون : يحب محمد س إذا كان يحب علي س

إذا أعطينا برولوق هدف (goal) أن يستخدم قوته الاستنتاجية لإيجاد كل الحلول الممكنة لإستخراج كل اللذين يحبون العدس.

بناء على الحقيقة أن علي يحب العدس وبناء على القانون أن محمد يحب كل ما يحبه علي فإن برنامج برولوق سوف تستنتج أن محمد يحب العدس وعليه فإن المخرجات ستكون :

علي " مخرج أصيل من الحقيقة يحب علي العدس "

محمد " مخرج استنتاجي من القانون يحب محمد س إذا كان يحب علي س.

هذا المثال يبدو بسيطاً جداً ولكن إذا كان هناك آلاف الحقائق فإن النتيجة أو الهدف سيكون معقداً جداً أو يصعب الوصول إليه بالطريقة اليدوية أو الذهنية كما أنه سيحتاج إلى برنامج معقد جداً إذا تم استخدام اللغات التقليدية .

٧. بعض مميزات لغة برولوق :

أ] التحكم التلقائي :

يتم التحكم في برنامج برولوق تلقائياً فعند تشغيل البرنامج يقوم النظام بالبحث عن كل القيم التي تحقق الهدف أثناء ذلك ربما يطلب الحاسوب إدخال بعض البيانات أو يعرض بعض المخرجات على الشاشة. هذه الخاصية تشترك فيها كل لغات البرمجة تقريباً.

ب] الخاصية الإستنتاجية :

هذه الخاصية أوضحناها في المثال السابق عندما أجاب برنامج برولوق بأن محمد يحب العدس بالرغم من أن هذه الحقيقة لم تكن ضمن الحقائق المبينة في البرنامج ولكن البرنامج إستنتجها بواسطة العلاقة يحب محمد س إذا كان يحب علي س.

ج] الخاصية الاسترجاعية :

ونعني بالخاصية الاسترجاعية عند وصول البرنامج لأي حل فإنه يستخدم هذا الحل في إعادة تصميم كل الفروض والعلاقات السابقة لمعرفة مدى تأثير هذا الحل في استنتاج حلول جديدة، أو كأن البرنامج يقوم بإعادة التنفيذ مرة أخرى مستخدماً الحل الذي توصل إليه كمدخل جديد في البرنامج. مثال ذلك إذا أضفنا في المثال السابق العلاقة أو القانون يحب عثمان ص إذا كان يحب محمد ص من قبل القانون يحب محمد س إذا كان يحب علي س فإنه عند تشغيل البرنامج إذا سألنا ماذا يحب عثمان فإن البرنامج سيجيب الفول مستخدماً حقيقة أن محمد يحب الفول ولكن عندما يصل إلى القانون يحب محمد س إذا كان يحب علي س فإنه سيصل إلى الحل أن محمد يحب العدس، ثم مستخدماً هذا الحل مرة أخرى في البرنامج راجعاً من البداية سيصل إلى الحل عثمان يحب العدس كذلك.

د] الأسلوب التخاطبي :

يمكن إستخدام لغة برولوق في إدخال البيانات بلوحة المفاتيح واستخراج المخرجات على الشاشة أثناء تنفيذ البرنامج وهذا يتيح التخاطب بين مستخدم البرنامج والحاسوب مما له من ميزة عملية في برامج الألعاب.

هـ] بنائية المعاودة :

تتميز لغة برولوق ببنائية المعاودة ونعني ببنائية المعاودة بأن ينادي كل قوس القوس الذي دونه أو بداخله

(مثال ٣) :

محمد (يدرس(الجامعة(العلوم (الحاسوب))))

تعني أن محمداً يدرس بالجامعة كلية العلوم بقسم الحاسوب. فإذا أضفنا

- علي يدرس (الجامعة (الآداب (الجغرافيا)))
- عثمان (يدرس (الجامعة (العلوم (الفيزياء))))
- أحمد (يدرس (الثانوية))

ثم سألنا من يدرس بالجامعة ستكون الإجابة

محمد

علي

عثمان

وإذا سألنا من يدرس بالعلوم ستكون الإجابة

محمد

عثمان

وإذا سألنا من يدرس جغرافيا ستكون الإجابة

علي

و] بساطة الأسلوب :

تتميز البرمجة بلغة برولوق بسهولة فهمها وقلة سطورها لأنها تستخدم لغة عادية غير اللغة الخوارزمية التي لاحظناها في لغة بيسك كما أن برنامج برولوق أقل عشرة مرات في عدد الأسطر من اللغات التقليدية تقريباً.

(مثال ٤) :

ذكرنا أن برنامج برولوق يتكون من أربع أجزاء. الجزء الأول ويعرف بالمجال (domain) ويتم فيه تعريف الأشياء التي يتعامل معها البرنامج والجزء الثاني ويعرف بالتحققات (predicates) ويتم فيه تعريف العلاقات التي تربط بين هذه الأشياء أما الجزء الثالث ويعرف بالعبارات (clauses) يتم فيه وصف الحقائق أو الواقع الحقيقي أما الجزء الرابع فيتم إعطاء التساؤلات التي يجيب عليها البرنامج أو ما يعرف بالأهداف(Goals) .

في هذا المثال نريد أن نعرف الرغبات الرياضية للطلاب حتى نستطيع أن ننظم النشاط الرياضي للطلاب بناء على ذلك. يبدأ البرنامج بتعريف الأشياء التي يتعامل معها البرنامج وهي طالب ونشاط رياضي، ثم تعريف العلاقة وهي رغبة الطالب في النشاط الرياضي المعني وهو كرة القدم أو كرة السلة أو كرة اليد أو كرة الطاولة ثم بعد ذلك يبدأ البرنامج في الإجابة من يرغب كرة القدم ومن يرغب كرة السلة وهكذا.

البرنامج **المجال** **طالب ، نشاط= رمزي** **التحققات** **يرغب (طالب ، نشاط)** **العبارات** **يرغب (علي، "كرة القدم")** **يرغب (عمر، "كرة السلة")** **يرغب (عثمان، "كرة اليد")** **يرغب (أحمد، "كرة الطاولة")** **يرغب (أحمد ، س) إذا كان يرغب (عمر، س)**	**/Program/*** **domains** **student, activity = symbol** **predicates** **likes (student, activity)** **clauses** **likes (ali, "foot ball").** **likes (omer, "basket ball").** **likes (osman, "volly ball").** **likes (ahmed, "ping pong").** **likes (ahmed ,X) if likes (omer , X).**

هذه العبارة تصف حقائق وهي رغبات مباشرة لهؤلاء الطلاب وقوانين (قانون واحد هنا) وهو أن أحمد يحب كل ما يحبه عمر.

الجزء الرابع وهو جزء التساؤلات يبدأ بكلمة هدف (goal) على النحو التالي:-

هدف **يرغب (عمر، "كرة السلة")** **نعم** **هدف** **يرغب (عمر،" كرة الطاولة")** **لا** **هدف** **يرغب (أحمد، "كرة السلة")** **نعم**	**Goal** **likes(omer,"basket ball").** **yes** **Goal** **likes(omer,"ping pong").** **No** **Goal** **likes(ahmed,"basket ball").** **yes**

نلاحظ في هذا البرنامج أولاً أن عبارات الحقائق والتساؤلات عن الحقائق تنتهي بنقطة في آخر العبارة أما التحققات فلا تنتهي بنقطة .

عندما سألنا البرنامج هل يرغب عمر لعبة كرة السلة كانت الإجابة ب "نعم" لأن ذلك ثابت في عبارات الحقائق، وكذلك عندما سألنا البرنامج هل يرغب أحمد كرة السلة؟ أجاب ب"نعم" مستنتجاً ذلك من حقيقة أن أحمد يرغب في كل ما يرغب فيه عمر من نشاط.

أما عندما سألنا البرنامج هل يحب عمر كرة الطاولة أجاب ب"لا" لأنه لا توجد عبارة تفيد بذلك أو علاقة يمكن أن يستنتج منها ذلك.

٨. المتغيرات في لغة برولوق:

الحرف X في البرنامج السابق والذي يقابله س في الترجمة العربية هو رمز لمتغير في لغة برولوق ونعني بالمتغير الرمز الذي يأخذ قيماً مختلفة فإذا كتبنا في المثال السابق :

هدف **يرغب (أحمد ، س)** **س=كرة الطاولة** **س=كرة السله**	**goal:** **likes (ahmed, X).** **X= ping pong** **X= basket ball** **2 solutions**

أي أن س أو (X) تحقق القيم كرة الطاولة وكرة السلة. القيمة الأولى من الحقيقة أن أحمد يرغب كرة الطاولة والقيمة الثانية مستنتجة بأن أحمد يحب كل نشاط يحبه عمر.

تشترط لغة برولوق في المتغيرات أن تبدأ بحرف كبير (capital) يليها أي عدد من الحروف والأرقـــام وعلامة (_) مثلاً My_name_ is تقبل كرمز لمتغير في لغة برولوق أما الرمز 1st- boy فلا تقبل لأنها بدأت برقم بدلاً من حرف كبير.

يفضل في لغات البرمجة عموماً أن تكون أسماء أو رموز المتغيرات ذات دلالة ومعنى بدلاً من س و ص و ع وغيرها مثلاً:

- **يفضل likes (ahmed , Sport) على likes (ahmed, X)**
- **يفضل likes (Student," foot ball") على likes (Y, "foot ball")**

إذا وضعنا هذه العبارة كهدف فإن البرنامج سيرد أن المتغير الذي يحقق الرغبة لكرة القدم هو فقط الطالب علي و يمكن صياغة هذا الهدف بلغة برولوق على النحو التالي :

هدف يحب (الطالب، كرة القدم) الطالب=علي حل واحد	goal: likes(student, foot ball). Student = ali I solution

٩. الأشياء والعلاقات في لغة برولوق :

ذكرنا أن جزء العبارات في لغة برولوق يتكون من حقائق (facts) والحقيقة يعبر عنها في شكل علاقة تربط بين أشياء. مثلاً في المثال السابق علي وعمر وعثمان وأحمد وكرة قدم وكرة سلة وكرة طاولة وكرة يد كلها شيئيات أما يرغب (likes) فهي علاقة. كما كان هناك شرط في أسماء المتغيرات هناك شرطاً لأسماء الشيئيات والعلاقات وهو أن يبدأ الاسم بحرف صغير (small) ثم أي تركيب من الحروف والأرقام وعلامة (_) ، وان لا يحتوي هذا التركيب علي مسافة فارغة .

معلومـــــة

يمكن استخدام الحروف الكبيرة أو المسافات الفارغة بين الحروف عندما ينحصر التركيب بين علامتي اقتباس " ".

(مثال ٥):

بين أي من العبارات أو الحقائق التالية صحيح أو خطأ :

فارهة (شقة)	`valuable (flat)`
فارهة (عربة)	`valuable (car)`
يمتلك (محمد، منزلاً)	`owns (mohemed, house)`
يأكل (القط، اللحم)	`eats (cat, meat)`
يأكل (البقر، النجيلة)	`eats (cow , grass)`
عربة (كرونا، زرقاء، موديل ٨٦)	`car(corona ,blue,M 86)`

كل هذه الأمثلة عبارات أو حقائق صحيحة في لغة برولوق ما عدا العبارة الأخيرة لأن **M** حرف كبير والأشياء يجب أن تبدأ بحرف صغير.

يلاحظ هنا أن عدد الأشياء داخل العلاقات مفتوح ونجد ان في :

- المثال الأول والثاني هناك شيء واحد يوصف بأنه فارهة .
- المثال الثالث هناك علاقة بين شيئين هما: المالك والمملوك، ومثله المثال الرابع والخامس حيث نجد علاقة بين شيئين هما: الآكل والمأكول .
- المثال السادس فهو وصف لعربة يتكون من نوع العربة، ولون العربة، وموديل العربة، وبالطبع يمكن أن يزيد الوصف ليشمل سعر العربة، ورقم العربة، وعدد الأميال التي قطعتها وهكذا.

١٠. المجالات والتحققات :

ذكرنا أن المجالات والتحققات في لغة برولوق يقابلان جزء التعريفات في لغة باسكال لتبيان ذلك نقدم الأمثلة التالية :

المجالات شخص ، نشاط = رمزي التحققات يحب (شخص ، نشاطاً)	Domains person , activity = symbol predicates likes (person, activity)

هذا يعني أن هناك علاقة بين شيئين هما الفرد والنشاط والعلاقة هي الحب أو الرغبة وهما من النوع رمز وليس من العدد أو الرقم.

(مثال ٦):

بين مصدر الخطأ إذا كتبنا الحقيقة (fact) التالية في جزء العبارات:

```
likes(10,x)
```

لن تقبل لأن الشيئ الأول في العلاقة معرف بأنه رمز وفي هذا المثال =١٠ وهي رقم ولذا سيعطي الحاسوب على الشاشة رسالة (type error) أي هناك خطأ أو عدم تطابق بين الأشياء ومواصفاتها.

(مثال ٧):

بين مصدر الخطإ إذا كتبنا الحقيقة (fact) التالية في جزء العبارات:

```
likes(ali,"foot ball","basket ball").
```

فإن البرنامج يعطي خطأ لأن العلاقة صممت بين شيئين اثنين فقط وهذه علاقة بين ثلاثة أشياء .

١١. الأهداف المركبة :

(مثال ٨):

شراء العربة المناسبة :

```
domains
   brand , colour = symbol
   age , price  = interger
   millage  = real
predicates
   car (brand, millage, age, colour, price)
clauses
   car (toyota, 45000,5, white, 800000) .
   car (ford, 30000,4, blue, 500000) .
   car (datsun, 20000, 3, red, 300000).
```

المجالات

الموديل ، اللون = رمزي

العمر ، السعر = رقمي

المسافة = عددي

التحققات

عربة(موديل، المسافة، العمر، اللون، السعر)

العبارات

عربة(تويوتا، ٤٥٠٠٠، ٥، أبيض، ٨٠٠٠٠٠)

عربة (فورد، ٣٠٠٠٠، ٤، أزرق، ٥٠٠٠٠٠)

عربة(داتسون، ٢٠٠٠٠، ٣، أحمر، ٣٠٠٠٠٠)

هدف

عربة(موديل، المسافة، العمر، اللون، ٥٠٠٠٠٠)

```
goal
car (Brand, Millage, Age, Colour, 500000)
```

الهدف الآن أن يبحث الحاسوب أو برنامج برولوق عن أي عربة سعرها ٥٠٠٠٠٠ وبغض النظر عن موديلها أو المسافة التي قطعتها أو عمرها أو لونها. سيجيب الحاسوب أن العربة التي تحقق ذلك هي العربة الفورد ومواصفاتها أن مسافتها ٣٠٠٠٠ ميل وعمرها ٤ سنوات ولونها أزرق .

```
Brand =ford Millage = 30000, Age =4m,
Colour = blue
1 solution
```

لكن عادة لا يبحث شخص عن عربة بسعر معين ولكن يبحث عن عربة بسعر لا يزيد عن مبلغ معين . لهذا سيعدل الهدف للآتي:

هدف
عربة (أي موديل، أي مسافة، أي عمر، أي لون، أي سعر) والسعر أقل من ٣٥٠٠٠.

```
goal
car ( Brand , Milage, Age, Colour, Price)
and Price < 35000.
```

ستكون الإجابة على هذا الهدف هي عربة الداتسون لأنها العربة الوحيدة التي تحقق شرط السعر أقل من ٣٥٠٠٠٠ وسيخرج البرنامج البينات التالية :

```
Brand = datsun , Millage = 20000, Age = 3,
Colour = red, Price=300000.
1 solution
```

إذا كنا لا نرغب في كل هذه البيانات عند إخراج الحل بل فقط نريد على سبيل المثال عمر العربة إضافة للسعر يمكن ذلك بتعديل الهدف على النحو التالي:

```
goal
Car (_,_,Age,_,Price )and Price <350000
Age = 3, price = 200000
```

من هذا نلاحظ أننا إذا وضعنا علامة (_) في مكان متغير يعني هذا أن ذلك المتغير غير مطلوب. وهذه العلامة إذا أستخدمت في الحقائق تعني أن الشئ الذي وضعت في مكانه العلامة شئ عام مثلاً .

(مثال ٩) :

ماذا تعني الحقائق التالية

١	**يرغب (_ ، كرة القدم)**	`likes (_ , foot all)`
٢	**يأكل (_ ، خبزاً)**	`eats ( _ , bread)`
٣	**يغسل (_)**	`washes( _ )`
٤	**يمتلك (_ ، ملابساً)**	`owns (_, clothes)`

١- يعني ذلك أن كل الطلاب يحبون كرة القدم .

٢- يعني أن كل الناس يأكلون الخبز .

٣- يعني أن كل الناس يغسلون ملابسهم .

٤- يعني أن كل شخص يمتلك ملابساً وهكذا .

(مثال ١٠):

إختيار زوج المباراة :

المجالات	`domains`
تلميذ = رمزي	`pupil = symbol`
عمر = رقمي	`age = integer`

التحققات	predicates
اللاعب (تلميذ ، عمر)	player (pupil , age).
العبارات	clauses
اللاعب (سامي، ٩)	player (sami, 9) .
اللاعب (عصام، ١٠)	player (isam, 10).
اللاعب (عمر ، ٩)	player (ali, 9).
اللاعب (على ، ٩)	player (omer, 9).
اللاعب (أحمد، ١٠)	player (Ahmed, 10).

فإذا أردنا من البرنامج إخراج كل تلميذين في عمر واحد ليتبارا في لعبة معينة فإن ذلك متاح في لغة برولوق على النحو التالي :

```
player (Pupil1,9) and player(Pupil2,9) and
Pupil2<>pupil1
```

اللاعب (التلميذ الأول ، ٩) واللاعب (التلميذ الثاني ، ٩) بحيث لا يكون التلميذ الأول هو نفس التلميذ الثاني.

سيقوم البرنامج بالبحث عن التلميذ الأول الذي عمره تسع سنوات وسيجد أول من يحقق ذلك التلميذ سامي ثم يبحث عن التلميذ الثاني الذي عمره تسع سنوات وسيجد أول من يحقق ذلك كذلك التلميذ سامي ولكن سيجد أن سامي لا يحقق الشرط الثالث وهو ألا يكون التلميذ الأول هو نفس التلميذ الثاني لذا سيقوم البرنامج بالبحث عن تلميذ آخر غير سامي وسيجد أول من يحقق شرط العمر لتسع سنوات ويختلف عن سامي هو التلميذ علي بهذا يكون علي هو أحد الحلول ثم يواصل البرنامج البحث عن تلميذ آخر يحقق شرط العمر تسع سنوات ويختلف عن سامي وسيجد أن عمر يحقق هذين الشرطين بعد ذلك يبدأ البرنامج بتكرار الإختبارات التي عملها مع

سامي مع علي ثم مع عمر وستكون النتيجة في النهاية على النحو التالي:

```
Pupil1= sami , Pupil2= ali
Pupil1= sami , Pupil2= omer
Pupil1= ali  , Pupil2= sami
Pupil1= ali  , Pupil2=  omer
Pupil1= omer , Pupil2= sami
Pupil1= omer , Pupil2= ali
```

وإذا عملنا إختباراً آخراً وهو أن يلعب عمر ٩ مع عمر ١٠

```
Player ( Pupil1 ,9) and Player (Pupil2, 10)
```

ستكون الإجابة :

```
Pupil1= sami , Pupil2= ahmed
Pupil1= ali  , Pupil2= ahmed
Pupil1= omer , Pupil2= ahmed
```

(مثال ١١):

برنامج الشجرة العائلية وصلة القرابة :

تتكون العائلة من ذكور وإناث ثم هناك قرابات مباشرة هي الأب والأم والجد والجدة والعم والعمة والخال والخالة والأخ والأخت والقرابة دائماً بين شخصين فقرابة الأبوة بين الأب وولده وقرابة الخالة بين الخالة وأبن أختها أو بنت أختها وهكذا. يمكن تفصيل هذه العلاقات لوصف الشجرة العائلية بلغة برولوق على النحو التالي:

```
/*family program */                    /* برنامج شجرة العائلية */
domains                                 المجالات
person = symbol                         شخص = رمزي
```

```
predicates                                  التحققات
male (person)                               ذكر (شخص)
female (person)                             أنثى (شخص)
mother (person, person                      أم (شخص و شخص)
father (person, person)                     أب (شخص وشخص)
brother (person, person)                    أخ (شخص وشخص)
sister (person, person)                     أخت (شخص وشخص)
uncle (person, person)                      عم (شخص وشخص)
aunt (person, person)                       عمة (شخص و شخص)
uncle1 (person, person)                     خال (شخص وشخص)
aunt1 (person, person)                      خالة (شخص و شخص)
grand_father(person, person)                جد (شخص و شخص)
grand_mother(person, person)                جدة (شخص و شخص)
parent (person, person)                     والد (شخص وشخص)
clauses                                     العبارات
male (ali).                                 ذكر (علي)
male ( gafer).                              ذكر (جعفر)
male (bakri).                               ذكر (بكري)
male(eisa).                                 ذكر (عيسى)
female (batol).                             أنثى (بتول)
female (fatma).                             أنثى (فاطمة)
female (maria).                             أنثى (ماريا)
female (samia).                             أنثى (سامية)
mother (samia, batol).                      أم (سامية ، بتول)
mother (ali , samia).                       أم (علي ، سامية)
father (ali, bakri).                        أب (علي ، بكري)
father (batol, gafar).                      أب (بتول ، جعفر)
father (maria, ali).                        أب (ماريا ، علي)
father (fatma, bakri).                      أب (فاطمة ، بكري)
```

```
parent(X,Y) if mother( X,Y) or father(X,Y ).

brother(X,Y) if male(Y) and parent(X,P) and
parent(Y,P) and X<>Y.

sister(X,Y) if female(Y) and parent(X,P) and
parent(Y,P) and X<>Y.

uncle(X,Y) if male(Y) and father(X,P) and
brother(Y,P).

aunt(X,Y) if female(Y) and father(X,P) and
brother(y,p).

uncle1(X,Y) if male(Y) and mother(X,P) and
sister(Y,p).

aunt1(X,Y) if female(y) and mother(X,P) and
sister(y,P).

grand_father(X,Y) if parent(X,P) and
father(P,Y).

grand_mother(X,Y) if parent (X,P) and
mother(P,Y).
```

والد (س ، ص) إذا كان أم (س، ص) أو أب (س، ص).

أخ (س، ص) إذا كان ذكر (ص) ووالد (س،ع) ووالد (ص،ع) و س تختلف عن ص.

أخت (س، ص) إذا كان أخت (ص) ووالد (س ،ع) ووالد (ص، ع) و س تختلف عن ص.

عم (س، ص) إذا كان ذكر (ص) وأب (س ،ع) وأخ (ص، ع) .
عمة (س، ص) إذا كان أخت (ص) وأب (س ،ع) وأخ (ص، ع).
خال (س، ص) إذا كان ذكر (ص) وأم (س ،ع) وأخت (ص، ع).
خالة (س، ص) إذا كان أخت (ص) وأم (س ،ع) وأخت (ص، ع).
جد (س، ص) إذا كان والد (س ،ع) وأب (ص، ع).
جدة (س، ص) إذا كان والد (س ،ع) أم (ص، ع).

إذا بدأنا في الإستفسار عن بعض العلاقات سيرد البرنامج حسب ماهو متاح له من معلومات ثم تعريفها له في جزء العبارات مثلاً

هدف: **Goal :**

جد (ماريا ، الجد) **Grand_father(maria , Gf)**
الجد = بكري **Gf = bakri**
حل واحد **1 solution**

أجاب البرنامج بأن جد ماريا هو بكري ولا يوجد جد آخر حسب المعلومات المتاحة لدى البرنامج .

هدف: **Goal:**
أخت (علي ، الأخت) **sister (ali , Sis)**
الأخت = فاطمة **Sis = fatima**
حل واحد **1 solution**

أجاب البرنامج بأن أخت علي هي فاطمة ولا توجد أخت أخرى لعلي حسب ما هو متاح من معلومات للبرنامج .

معلومـــــة

١. تتيح لغة البرولوق إمكانية كتابة الملاحظات أو التعليقات إذا :
- وضعت بين علامتي /* */ .
- وضعت علامة % عند بداية السطر وتنتهي الملاحظة أو التعليق بنهاية السطر.
- يتبع الهدف (Goal) علامة (:)

(مثال ١٢)

برنامج التوصل إلى القاتل :

لقد تم اغتيال سهام في مطعم بالسوق، وقد شوهد بعض الأفراد بعد عملية القتل مباشرة، وبدأت الشرطة في التحري لمعرفة القاتل . فجمعت معلومات عن الوسيلة التي يمكن أن تكون قد أستخدمت في تنفيذ عملية القتل، وهل هذه الوسيلة يمكن أن تكون في يد أيٍ من المشبوهين. من جانب آخر تحرت الشرطة عن علاقات سهام بالمشبوهين، وهل هناك دوافع لأي من المشبوهين لقتل سهام وحيث أن هذه المعلومات يمكن أن تكون شائكة ومعقدة .

تم أستخدم الحاسوب للمساعدة في الوصول للقاتل وكانت اللغة المناسبة التي يمكن التخاطب بها مع الحاسوب هي لغة برولوق ولهذا الغرض قام الضابط الشرطي بكتابة البرنامج التالي:

```
domains
name = symbol
age= integer
sex = symbol
motive = symbol
mark  = symbol
desine = symbol
activity = symbol
predicates
person (name, age, sex, activity)
interbles (name, name )
killed (name, weapon)
marked_with (name, mark)
motive (desire)
owns(name, weapen)
owns_proboly(name,weapen
operates_identically (weapen , weapen)
suspect (name)
clauses
person (ali,25, m,football_player).
person (ali, 25, m , putcher) .
person (fatma , 20, f, hair_dresser).
person (ammar, 25, m, carpenter) .
person (fatih, 25, m, pickpoket) .
introubles (fatma, fatih) .
introubles (fatma , ammar).
introubles (siham, ammar).
killed with (siham, club).
```

```
motive (money).
motive (troubles).
marked (samia , blood).
marked (ali , mud).
owns (ammar , woodenleg).
owns (fatih , pistol) .

*/background - knowledge/*

operates_identically (woodenleg, club).
operntes_identically (club,cafeteria).
operntes_identically (scissors, knife).
operates_identically (boot, cafeteria).
owns_probably(X,boot) if
   person(X,-,-,football_player).
owns_probably(X,scissors) if
   person(X,-,-,hair_dresser).
owns_probably(X,object)if
   owns_probably(X,Object).

goal:
/* Suspect those who own aweapon with which
Siham has been killed */

suspect(X) if killed(Siham,weapon)and
operates_identically(Y,weapen) and
owns_probably(X,Y).

X=ammar
1 solution

/* suspect persons who have troubles with
Siham */
```

```
goal:
person(X, _, _,_) and introubles(siham ,
X).

X=ammar
1 solution

/* suspect kichpockets whose motive coud be
mony  */

goal:
 suspect(X) if motive(money) and
       person(X, _, _,pickpocket).

fatih
1 solution
```

أخرج البرنامج بأن المتهم الرئيس هو عمار للأسباب التالية :

أولاً : أن سهام قتلت بضربة عصا غليظة وهذا يشبه أن تكون قد ضربت بحائط الكافتريا أو بالبوت ولما كان علي يمتلك بوت ويمتلك عمار رجل خشبية من هذا كانا أكثر شبهة من الآخرين .

ثانياً : يمكن أن يكون الفاتح متهماً بالقتل بضرب سهام مع حائط الكافتريا بغرض سرقة مالها.

ثالثاً : بما أن هنالك إشكال بين عمار وسهام وأن سهام قد قتلت بعصا غليظة يكون أقرب المتهمين هو عمار.

المجالات

الإسم= رمزي

العمر= رقمي

النوع =رمزي

الدافع= رمزي

العلامة= رمزي

السلاح =رمزي

النشاط =رمزي

التحققات

شخص(الإسم ، العمر ، النوع، النشاط)

إشكال (الإسم ، الإسم)

قتل (الإسم ، السلاح)

علامة (إسم ، علامة)

دافع (الدافع)

يمتلك (الإسم ، السلاح)

يحتمل أن يملك (الإسم،السلاح)

يؤثر مثل الآخر (السلاح، السلاح)

المتهم (الإسم)

العبارات

شخص(علي، ٢٥ ، ذكر ، لاعب كرة).

شخص (علي، ٢٥، ذكر، جزار) .

شخص(فاطمة ، ٢٠ ، أنثى ، كوافير) .

شخص (عمار ،٢٥، ذكر، نجار).

شخص (فاتح، ٢٥، ذكر، نشال).

إشكال (فاطمة، فاتح).

إشكال (فاطمة، عمار).

إشكال (سهام، عمار).

قتل (سهام، عصا غليظة).

الدافع (المال).

الدافع (المشاكل).

علامة(سامية ، دم).
علامة (علي ، طين).
يمتلك (عمار ، رجل خشبية).
يمتلك (الفاتح ، مسدس) .

*/معلومات خلفية */
له نفس الأثر (رجل خشبية، كافتريا).
له نفس الأثر (كافتريا، قهوة).
له نفس الأثر(المقص ، المطواة).
له نفس الأثر (البوت، الكافتريا).
يمتلك (س، بوت) إذا كان شخص (س،-،-، لاعب كرة).
يمتلك (س، مقص) إذا كان شخص (س، -، -، كوافير).
يمتلك (س ، شيئاً) إذا كان يحتمل أن يمتلك (س، شيئاً).
هدف:
/*إتهم من يملك سلاحاً يمكن أن تكون قد قتلت به سهام*/
إتهم (س) إذا كان قتل (سهام، السلاح) وله نفس الأثر (ص، السلاح) ويحتمل أن يملك (س، ص)
س= عمار
حل واحد
/*إتهم أولئك الأشخاص الذين لديهم مشاكل مع سهام*/
هدف:
إتهم (س) إذا كان دافع (المشاكل) وشخص (س،-،-،-) وإشكال (سهام،س)
س= عمار
حل واحد

/إتهم النشالين الذين دافعهم المال/
إتهم (س) إذا كان دافع (المال) و شخص (س، -، -، نشال)

س=فاتح
حل واحد

١٢. تمرين

١/ ماعلم الذكاء الإصطناعى؟

٢/ ماذا نعنى بذكاء الآله؟

٣/ أعط مثالا عن ذكاء الآله فى التعرف على معتادى الإجرام.

٤/ أعط مثالا لذكاء الآله فى لعبة الشطرنج .

٥/ صف ذكاء الآله فى الانسان الآلى.

٦/ ما أهداف علم الذكاء الاصطناعى؟

٧/ ما أسس علم الذكاء الاصطناعى؟

٨/ صف مواصفات لغات الذكاء الاصطناعى.

٩/ ما أهم لغات الذكاء الإصطناعى؟ وأين تم تصميمها؟ ومتى؟ وما أهم مواصفاتها؟

١٠/ ماذا نعنى من أن لغات الذكاء الاصطناعى أكثر كفاءة من اللغات التقليديه وما الفرق بينهما؟

١١/ صف بنائية لغة برولوق.

١٢/ ما مميزات لغة برولوق؟

١٣/ أعط مثالا بلغة برولوق لتنظيم النشاط الرياضى.

١٤/ كيف يتم التعامل مع المتغيرات فى لغة برولوق ؟ (أعط أمثله) .

١٥/ صف الأشياء والعلاقات فى لغة برولوق (اعط أمثله).

١٦/ اكتب برنامجا بالعربية أو الإنجليزية يمكن الحاسوب من أعطاء خيارات لشراء عربه بمواصفات معينة فى حدود مبلغ معين .

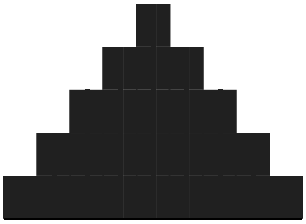
١٧/ اكتب برنامجا بالعربية أو الانجليزية يمكن الحاسوب من إختيار زوج المصارعه المناسب لطلاب المدرسه.

١٨/ اكتب برنامجا بالعربية أو الإنجليزيه يصف شجرة العائلة والقرابات.

١٩/ اكتب برنامجا بالعربية او بالانجليزية يكتشف القاتل فى جريمة قتل.
٢٠/ اذكر ثلاثة من تطبيقات الذكاء الإصطناعي.
٢١/ اكتب برنامجاً باللغة العربية أو الانجليزية للعبة برج هانوي التي تتكون من ثلاثة اعمده يحتوي العمود الأول علي عدد من الأقراص المختلفة القطر تكون مرتبة بحيث يكون الأصغر قطراً فوق الأكبر قطراً، والمطلوب هو نقل كل الأقراص الي أحد الأعمده الأخري مرتبة بنفس الترتيب شريطة الالتزام بالأتي:

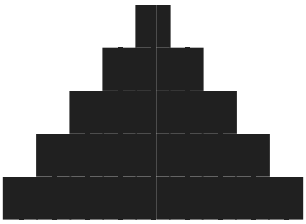
- نقل قرص واحد في المرة الواحده.
- لا يوضع قرص فوق اخر أصغر قطراً.

الوضع الأصلي



الأول الثاني الثالث

الوضع النهائي



الأول الثاني الثالث

٢٢/ يعتبر علم الذكاء الاصطناعي نافذه تطل علي المستقبل. ناقش هذه العباره.
٢٣/ قارن بين بنية برولوق وبينة أي لغة اخري درستها.

الملاحق

الملحق : عبارات التحكم في لغة بسكال

حلقة For ... Do:

نستخدم هذا النمط من الحلقات التكرارية إذا كان معلوماً لدينا مسبقاً عدد المرات التي نريد فيها تكرار العمل.

الصيغة العامة

```
For <متغير>:=<القيمة الابتدائية للمتغير> to[Downto]
                                    <القيمة النهائية للمتغير > do
        <عبارات بسيطة أو مركبة>
```

العبارة المركبة هي التي تتألف من عدد من العبارات البسيطة وينبغي أن تنحصر بين Begin . . . End ;

(مثال ١):

الحلقة التكرارية التالية تقوم بحساب مربع الأعداد من ١٠الي٣٠ :

```
For I:=10 to 30 do
    Writeln(I*I);
```

(مثال ٢):

الحلقة التكرارية التالية تقوم بحساب مربع الأعداد من٣٠ الي ١٠ :

```
For I:=30 downto 10 do
    Writeln(I*I);
```

يمكن أن نستخدم العبارة المركبة في المثال السابق لتوضيحها فقط

```
For I:=30 downto 10 do
Begin
    X:=I*I;
    Writeln(X);
End;
```

(مثـــال ٣):

اكتب برنامجاً بلغة بسكال يقوم بحساب مجموع مربعات الأعداد من ١٠ إلى ٣٠ ثم الوسط الحسابي لها :

```
Var
      I, S : integer ;
Begin
      S:=0;
      For I:=10 to 30 do
           S:=S +I*I;
      Writeln(S,s/21);
End.
```

(مثـــال ٤):

اكتب برنامجاً بلغة بسكال يقوم بحساب مجموع مربعات الأعداد من a إلى b حيث a>b ثم الوسط الحسابي لها :

```
Var
      I, S,a,b : integer ;
      Avg : real;
Begin
      ReadLn (a, b);
      S:=0;
      For I:=a to b do
           S:=S +I*I;
      Avg:=S/(b-a+1);
      Writeln(' مجموع مربعات الأعداد = ',S);
      Writeln(' المتوسط = ',Avg);
End.
```

(مثـــال ٥):

اكتب برنامجاً بلغة بسكال يقوم بحساب مضروب أي عدد N

```
Var
      I,N: shortint ;
      Fact : Longint;
Begin
      Write('ادخل العدد');
      ReadLn (N);
      Fact :=1;
      For I:=N downto 1 do
           Fact:=Fact*I;
      Writeln(' مضروب العدد = ',Fact);
End.
```

حلقة Repeat ... Until:

نستخدم هذا النمط من الحلقات التكرارية إذا كان تكرار العمل يتوقف علي شرط معين .

الصيغة العامة

```
Repeat
     <أي تتابع من العبارات البسيطة أو المركبة>
Until <تعبير بولياني>
```

حلقة While ... Do:

نستخدم هذا النمط من الحلقات التكرارية إذا كان تكرار العمل يتوقف علي شرط معين .

الصيغة العامة

```
While <تعبير بولياني> do
     <عبارات بسيطة أو مركبة>
```

عبارة IF ... THEN ... ELSE :

```
IF <تعبير بولياني> Then
     <عبارات بسيطة أو مركبة>
Else
     <عبارات بسيطة أو مركبة>
```

(مثـــال ٦):

اكتب برنامجاً بلغة بسكال يقوم بطباعة "نجاح" اذا كانت درجتة أكبر من أو تساوي ٥٠ وإلا يطبع "رسوب"

```
Var
     score: shortint ;
Begin
     Write('إدخل درجة الطالب');
     Readln(score);
     If score>=50 then
          Writeln('نجاح')
     Else
          Writeln('رسوب');
 End.
```

(مثـــال ٧):

أكتب برنامجاً بلغة بسكال يقوم بطباعة "حرف علة " إذا كان الحرف المدخل أحد حروف العلة "A,E,I,O,U" وإلا يطبع "ليس حرف علة"

```
Var
      C: Char ;
Begin
      Write('أدخل أي حرف');
      Readln(C);
      C:=Upcase(C);
      If (C='A')OR (C='E')OR (C='I')OR
           (C='O')OR (C='U')OR  then
           Writeln('حرف علة')
      Else
           Writeln('ليس حرف علة');
 End.
```

يمكن استخدام عبارة IN (ينتمي)المنطقية مع عبارة If ليصبح حـــل المثـــال السابق علي النحو التالي

```
Var
      C: Char ;
Begin
      Write('أدخل أي حرف');
      Readln(C);
      C:=Upcase(C);
     If C IN ['A','E','I','O','U'] then
           Writeln('حرف علة')
      Else
           Writeln('ليس حرف علة');
 End.
```

(مثـــال ٨):

اكتب برنامجاً بلغة بسكال يقوم بطباعة "كسلا " أو "جوبا" أو "مدني" أو "الفاشر" أو "دنقلا" إذا كان الرقم المدخل هو ١ أو ٢ أو ٣ أو ٤ أو ٥ علي الترتيب .

```
Var
     x: Byte ;
Begin
     Write('أدخل رقم من ١ الي ٥ ');
     Readln(x);
     If not x IN[1..5] then
          Writeln("الرقم المدخل خارج المدى المحدد");
     Else
     Begin
          If (x=1) then Writeln('كسلا');
          If (x=2) then Writeln('جوبـا');
          If (x=3) then Writeln('مـدنـي');
          If (x=4) then Writeln('الـفـاشر');
          If (x=5) then Writeln('دنـقـلا');
     End;
End.
```